UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TELEMÁTICA
INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

"Diseño e implementación de un terminal de control y monitorización para una centralita basada en *Asterisk*."

AUTOR: Julia Koblents Lápteva
TUTOR EMPRESA: D. JOSÉ MANUEL RUIZ DE MARCOS
TUTOR UC3M: D. MARIO MUÑOZ ORGANERO

PROYECTO FIN DE CARRERA

Título: Diseño e implementación de un terminal de control y monitorización para una centralita basada en *Asterisk*.
Autor: Julia Koblents Lápteva
Tutor Empresa: D. José Manuel Ruiz de Marcos
Tutor UC3M: D. Mario Muñoz Organero
Departamento: Ingeniería Telemática

MIEMBROS DEL TRIBUNAL

Presidente:
Secretario:
Vocal:

FECHA DE LECTURA:

CALIFICACIÓN:

FIRMAS:

Presidente **Secretario** **Vocal**

RESUMEN

Realizando una breve reflexión sobre el estado del mundo de las telecomunicaciones en general, se puede observar a lo largo de las últimas décadas una constante y abrumadora evolución. El segmento dedicado a la telefonía no es una excepción y tras la convergencia de las redes de voz y de datos, se ha escrito un nuevo capítulo en la historia con la creación de la *VoIP*. Esta tecnología ha posibilitado la introducción de mejoras impensables hasta el momento y se puede considerar que ha supuesto el inicio de una nueva era.

Uno de los acontecimientos importantes dentro de dicho proceso fue la aparición en el mercado de centralitas software que permitían disminuir de forma vertiginosa el coste de las instalaciones telefónicas para las pequeñas y medianas empresas. Además, debido al modelo del desarrollo del mencionado software, su perfeccionamiento es constante, al igual que el aumento de las funcionalidades introducidas.

Uno de los pioneros en este campo de investigación fue el creador de *Asterisk*, aunque a día de hoy ya existen numerosos competidores que pretenden ofrecer un producto sustitutivo o al menos complementario.

Sin embargo, los potenciales clientes de las centralitas telefónicas descritas, siguen siendo reticentes a apostar por una tecnología que consideran inmadura. A pesar de las ventajas presentadas, prefieren confiar en marcas de renombre que apostar por la innovación.

Es por tanto imprescindible establecer una diferenciación para este producto software para ofrecer al cliente no sólo un producto sustitutivo sino uno mucho más completo. Bajo esta filosofía nace la necesidad de desarrollo de productos complementarios para las centralitas, que permitan ofrecer servicios impensables en las centralitas tradicionales y conseguir así presentar una competencia real a los productos asentados en el mercado.

Dentro de esta tendencia se planteó el desarrollo del producto cuyo proceso de realización se describe en la presente memoria, y que consiste en la implementación de una herramienta que permite una monitorización activa y control de una centralita basada en *Asterisk*. Es decir, se trata de una aplicación web que proporciona un interfaz con información en tiempo real sobre el estado de los diversos componentes del sistema. A su vez permite al usuario encargado de la monitorización interactuar con la centralita poniendo a su disposición diversas funcionalidades con el propósito de facilitar y complementar el trabajo de éste.

Como resultado de la realización de este proyecto se ha obtenido un producto que satisface ciertas demandas del mercado ya que consigue un complemento de las centralitas que permite una supervisión de los usuarios, una distribución efectiva de las llamadas y otras numerosas funcionalidades.

Debido a la constante evolución del software de *Asterisk* y la incesante aparición de nuevas utilidades, también es posible adaptar y perfeccionar el producto desarrollado acorde a las innovaciones conseguidas.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN Y OBJETIVOS *8*
1.1. Introducción. 8
1.2. Objetivos del Documento. 8
1.3. Estructura del Documento. 8
1.4. Fines del proyecto. 9
1.5. Fases del Proyecto. 10
1.5.1. Fase I: Investigación previa y Estado del arte. 10
1.5.2. Fase II: Estudio detallado del API de *Asterisk Manager*. 10
1.5.3. Fase III: Estudio de una *PBX Asterisk*. 10
1.5.4. Fase IV: Diseño y desarrollo del software del producto. 11
1.5.5. Fase V: Desarrollo de la interfaz del producto. 11
1.5.6. Fase VI: Pruebas finales del producto. 12
1.5.7. Fase VII: Puesta en marcha. 12
1.5.8. Fase VIII: Documentación del proyecto. 12
2. ESTADO DEL ARTE *14*
2.1. Introducción. 14
2.2. Descripción de *PSTN* y comparación con los servicios *VoIP*. 14
2.2.1. El principio de las *PSTN*. 14
2.2.2. Entendiendo las modernas *PSTN*. 15
2.2.2.1. Analógico vs Digital. 15
2.2.2.2. PCM. 15
2.2.2.3. Bucles Locales, Troncales y Comunicación entre los Conmutadores. 16
2.2.2.4. Servicios y Aplicaciones de la *PSTN*. 16
2.2.3. Factores influyentes en la convergencia de las redes de voz y de datos. 17
2.2.3.1. Inconvenientes de la PSTN. 17
2.2.3.2. Coste económico. 18
2.2.3.3. Características de redes IP. 18
2.3. Introducción a *Asterisk*. 19
2.4. Tecnologías similares a *Asterisk*. 20
2.4.1. FreeSWITCH. 21
2.4.2. SIP Express Router. 22
2.4.3. sipX ECS. 22
2.4.3.1. Aplicaciones. 23
2.4.3.2. Hardware. 24
2.4.3.3. sipX vs Asterisk. 24
2.4.3.3.1. Estructura. 24
2.4.3.3.2. Características. 24
2.4.3.3.3. Calidad de voz. 25
2.4.3.3.4. Administración. 25
2.4.3.3.5. Apoyos. 25
2.4.4. YATE. 25
2.4.5. Otras alternativas a Asterisk. 26
2.5. Descripción de *Asterisk*. 27
2.5.1. Módulos Cargables. 27
2.5.2. Funciones de *Asterisk*. 29
2.5.3. Estructura de Asterisk. 30
2.5.4. *Dialplan*. 31
2.5.4.1. Contextos. 32
2.5.4.2. Extensiones. 32
2.5.4.3. Prioridades. 32
2.5.4.4. Aplicaciones. 33
2.5.5. *Asterisk* manager API. 33

2.5.5.1. Características del protocolo. 34
2.5.5.2. Tipos de paquetes. 34
2.5.5.3. *Asterisk* Manager Actions. 35
2.5.5.4. Asterisk Manager Events. 35
3. MEMORIA TÉCNICA 39
3.1. Introducción. 39
3.2. Herramientas utilizadas. 40
3.2.1. Herramientas software 40
3.2.2. Equipamiento hardware 48
3.3. Diseño. 48
3.3.1. Funcionalidad requerida. 49
3.3.2. Decisiones de diseño. 50
3.3.3. Arquitectura de la aplicación. 52
3.3.3.1. Módulo de autenticación. 52
3.3.3.2. Módulo de visualización. 53
3.3.3.2.1. Módulo de configuración del interfaz. 53
3.3.3.2.2. Módulo de control de contactos 54
3.3.3.2.3. Módulo de control de llamadas. 55
3.3.3.2.4. Módulo de control del estado. 55
3.3.3.2.4.1. Estado de las extensiones 55
3.3.3.2.4.2. Estado de las líneas. 57
3.3.3.3. Módulo de procesado de eventos. 57
3.3.3.4. Módulo de envío de acciones. 59
3.3.4. Especificación. 60
3.3.4.1. Módulo de autenticación. 60
3.3.4.2. Módulo de visualización. 63
3.3.4.2.1. Módulo de configuración del interfaz. 63
3.3.4.2.2. Módulo de control de contactos 66
3.3.4.2.3. Módulo de control de llamadas. 70
3.3.4.2.4. Módulo de control de estado. 72
3.3.4.2.4.1. Estado de las extensiones 72
3.3.4.2.4.2. Estado de las líneas. 74
3.3.4.3. Módulo de procesado de eventos. 76
3.3.4.4. Módulo de envío de acciones. 81
3.4. Configuración. 83
3.4.1. Configuración de *Asterisk*. 83
3.4.2. Configuración de las tarjetas. 84
3.4.3. Configuración de los teléfonos *IP*. 86
3.4.4. Configuración del Terminal de Operador 88
3.5. Pruebas realizadas. 88
3.5.1. Llamadas del sistema. 88
3.5.2. Establecimiento de llamadas mediante el terminal. 90
3.5.3. Transferencia de llamadas del sistema. 93
3.5.4. Transferencia de llamadas mediante el terminal. 94
3.5.5. Priorización de una llamada entrante. 96
3.5.6. Pruebas de monitorización. 98
3.5.7. Introducción de credenciales. 102
3.5.8. Configuración del interfaz. 103
3.5.9. Pruebas de estrés. 105
4. MANUAL DE USUARIO 108
4.1. Introducción. 108
4.2. Introducción de las credenciales de usuario. 108
4.3. Vistas de Operador. 109
4.3.1. Interfaz de Operador 110

4.3.2. Extensiones monitorizadas. 111
4.3.3. Agenda del usuario operador. 114
4.3.4. Llamadas en curso del sistema. 115
4.3.5. Líneas analógicas y digitales del sistema. 120
4.4. Configuración del Terminal de Operador. 123
4.5. Agenda. 132
5. MEMORIA ECONÓMICA Y PRESUPUESTO 136
5.1. Introducción. 136
5.2. Coste de materiales. 136
5.2.1. Coste del software empleado. 136
5.2.2. Coste del hardware empleado. 136
5.3. Coste de personal. 137
5.4. Otros gastos. 139
5.5. Coste total de ejecución. 139
6. CONCLUSIONES Y POSIBLES MEJORAS 141
6.1. Conclusiones. 141
6.2. Posibles mejoras. 142
7. REFERENCIAS Y BIBLIOGRAFÍA 147
7.1. Glosario. 147
7.2. Referencias. 150
7.3. Bibliografía. 152

ÍNDICE DE FIGURAS

FIGURA 1.1: EJEMPLO DE LA VISUALIZACIÓN DEL TERMINAL DE OPERADOR. 12
FIGURA 2.1: ILUSTRACIÓN DE UNA RED MALLADA. 14
FIGURA 2.2: ALTERNATIVAS A *ASTERISK*. 21
FIGURA 2.3: ESTRUCTURA DE *ASTERISK*. 29
FIGURA 3.1: ARQUITECTURA DE *J2EE*. 46
FIGURA 3.2: ARQUITECTURA DE LA APLICACIÓN "TERMINAL DE OPERADOR". 52
FIGURA 3.3: DIAGRAMA DE FLUJO DEL MÓDULO DE AUTENTICACIÓN. 61
FIGURA 3.4: GRAFO DE LLAMADA DE LA FUNCIÓN DE INICIALIZACIÓN DEL MÓDULO DE AUTENTICACIÓN. 62
FIGURA 3.5: DIAGRAMA DE FLUJO DEL MÓDULO DE CONFIGURACIÓN DEL INTERFAZ. 64
FIGURA 3.6: GRAFO DE LLAMADA DE LA FUNCIÓN DE INICIALIZACIÓN DEL MÓDULO DE CONFIGURACIÓN DEL INTERFAZ. 65
FIGURA 3.7: GRAFO DE LLAMADA DE LA FUNCIÓN *CONFIGURAR*. 66
FIGURA 3.8: GRAFO DE LLAMADA DE LA FUNCIÓN *SCREENSELECT*. 66
FIGURA 3.9: DIAGRAMA DE FLUJO DEL MÓDULO DE CONTROL DE CONTACTOS. 67
FIGURA 3.10: GRAFO DE LLAMADA DE LA FUNCIÓN *CLICKTOCALL*. 68
FIGURA 3.11: GRAFO DE LLAMADA DE LA FUNCIÓN *TRANSFERIR*. 69
FIGURA 3.12: GRAFO DE LLAMANTES DE LA FUNCIÓN *TRASFERIR*. 69
FIGURA 3.13: DIAGRAMA DE FLUJO DEL MÓDULO DE CONTROL DE LLAMADAS. 71
FIGURA 3.14: GRAFO DE LLAMADA DE LA FUNCIÓN *WRITE_PHONEIN*. 71
FIGURA 3.15: GRAFO DE LLAMADA DE LA FUNCIÓN *WRITE_PHONEOUT*. 72
FIGURA 3.16: DIAGRAMA DE FLUJO DE MÓDULO DE ESTADO DE LAS EXTENSIONES. 73
FIGURA 3.17: GRAFO DE LLAMADA DE LA FUNCIÓN *WRITE_EXTENSIONS*. 73
FIGURA 3.18: DIAGRAMA DE FLUJO DEL MÓDULO DE ESTADO DE LAS LÍNEAS. 75
FIGURA 3.19: GRAFO DE LLAMADA DE LA FUNCIÓN *WRITE_PORT*. 75
FIGURA 3.20: GRAFO DE LLAMADA DE LA FUNCIÓN *REDIRECTCALLLINE*. 76
FIGURA 3.21: DIAGRAMA DE FLUJO DEL MÓDULO DE PROCESADO DE EVENTOS. 77
FIGURA 3.22: REPRESENTACIÓN DEL ARCHIVO *JBOSS-SERVICE.XML*. 78
FIGURA 3.23: GRAFO DE LLAMADA DE LA FUNCIÓN *ONMESSAGE*. 79
FIGURA 3.24: GRAFO DE LLAMADA DE LA FUNCIÓN *COMPROBARRELEVANTE*. 80
FIGURA 3.25: GRAFO DE LLAMADA DE LA FUNCIÓN *REDRAWINFORMATION*. 80
FIGURA 3.26: DIAGRAMA DE FLUJO DEL MÓDULO DE ENVÍO DE ACCIONES. 81
FIGURA 3.27: INTERFAZ DE INICIO DE CONFIGURACIÓN DE UN TELÉFONO IP. 86
FIGURA 3.28: INTERFAZ DE CONFIGURACIÓN DE CARACTERÍSTICAS SIP. 87
FIGURA 3.29: INTERFAZ DE CONFIGURACIÓN DE CUENTA DE USUARIO. 87
FIGURA 3.30: LLAMADA ENTRANTE ATENDIDA. 89
FIGURA 3.31: INICIO DE UNA LLAMADA ENTRANTE AL SISTEMA. 89
FIGURA 3.32: LLAMADA ENTRANTE AL SISTEMA PERDIDA. 89
FIGURA 3.33: ILUSTRACIÓN DE UNA CONVERSACIÓN ENTRE DOS USUARIOS DEL SISTEMA. 90
FIGURA 3.34: ESTABLECIMIENTO DE UNA LLAMADA CON OTRO USUARIO DEL SISTEMA. 91
FIGURA 3.35: ESTABLECIMIENTO DE UNA LLAMADA CON UN CONTACTO REGISTRADO. 91
FIGURA 3.36: ESTABLECIMIENTO DE UNA LLAMADA CON UN NÚMERO NO REGISTRADO EN LA AGENDA. 92
FIGURA 3.37: INTENTO DE ENCAMINAMIENTO DE UNA LLAMADA POR LA LÍNEA SELECCIONADA. 92
FIGURA 3.38: ESTABLECIMIENTO DE UNA LLAMADA POR UNA LÍNEA EN CONCRETO. 93
FIGURA 3.39: INICIO DE UNA LLAMADA ENTRANTE AL SISTEMA. 93
FIGURA 3.40: EL USUARIO DESTINATARIO RESPONDE A LA LLAMADA ENTRANTE AL SISTEMA. 93
FIGURA 3.41: EL USUARIO DESTINATARIO PONE A LA LLAMADA ENTRANTE EN ESPERA. 94
FIGURA 3.42: EL USUARIO DESTINATARIO INTENTA CONTACTAR CON OTRO USUARIO DEL SISTEMA. 94
FIGURA 3.43: EL USUARIO DESTINATARIO HA TRANSFERIDO LA LLAMADA ENTRANTE A OTRO USUARIO DEL SISTEMA. 94
FIGURA 3.44: INICIO DE UNA LLAMADA ENTRANTE. 95

FIGURA 3.45: EL USUARIO OPERADOR SELECCIONA EL ORIGEN DE LA LLAMADA Y LO ARRASTRA HACIA OTRO USUARIO DEL SISTEMA 95
FIGURA 3.46: EL USUARIO OPERADOR SUELTA EL RATÓN Y SE REALIZA LA TRANSFERENCIA DE LA LLAMADA. 95
FIGURA 3.47: INICIO DE UNA LLAMADA SALIENTE DEL SISTEMA. 95
FIGURA 3.48: EL USUARIO ARRASTRA EL NÚMERO DESTINO HACIA UNA ENTRADA DE LA AGENDA. 96
FIGURA 3.49: COMO RESULTADO DE LA TRANSFERENCIA SE ESTABLECE UNA NUEVA LLAMADA ENTRE DOS USUARIOS QUE NO PERTENECEN AL SISTEMA. 96
FIGURA 3.50: GENERACIÓN DE DOS LLAMADAS ENTRANTES AL SISTEMA. 97
FIGURA 3.51: EL USUARIO OPERADOR DECIDE ATENDER LA SEGUNDA LLAMADA GENERADA. 97
FIGURA 3.52: EL USUARIO SE ENCUENTRA OCUPADO CON LA CONVERSACIÓN PRIORIZADA. 97
FIGURA 3.53: EL USUARIO OPERADOR REALIZA UN DOBLE CLICK SOBRE LA LLAMADA APARCADA. 97
FIGURA 3.54: EL USUARIO OPERADOR RECUPERA LA LLAMADA ANTERIORMENTE APARCADA. 98
FIGURA 3.55: EL USUARIO RETIENE UNA LLAMADA SALIENTE. 98
FIGURA 3.56: EL USUARIO RECUPERA LA LLAMADA SALIENTE ANTERIORMENTE RETENIDA. 99
FIGURA 3.57: UNA LLAMADA ENTRANTE PARA EL USUARIO U. OPERADOR. 99
FIGURA 3.58: LA LLAMADA ENTRANTE ES RECOGIDA POR EL USUARIO R. GUTIÉRREZ. 99
FIGURA 3.59: VISTA GENERAL DE LA AGENDA DEL USUARIO. 100
FIGURA 3.60: RESULTADO DE LA BÚSQUEDA DEL PATRÓN "FIJ". 100
FIGURA 3.61: RESULTADO DE LA BÚSQUEDA DEL PATRÓN "JIMENEZ". 101
FIGURA 3.62: INGRESO DEL USUARIO A LA SALA DE CONFERENCIAS NÚMERO 2. 101
FIGURA 3.63: ACCESO AL SISTEMA DEL USUARIO OPERADOR. 102
FIGURA 3.64: VISTA DEL INTERFAZ DE INICIO DEL USUARIO OPERADOR. 103
FIGURA 3.65: PARTE DE LA CONFIGURACIÓN REFERENTE A LA PARRILLA DE LAS EXTENSIONES. 104
FIGURA 3.66: PARTE DE LA CONFIGURACIÓN REFERENTE A LA IDENTIFICACIÓN DE USUARIOS Y SELECCIÓN DE PANTALLAS. 104
FIGURA 3.67: TERMINAL DE OPERADOR CORRESPONDIENTE A LA CONFIGURACIÓN REALIZADA. 105
FIGURA 3.68: GENERACIÓN DE VARIAS LLAMADAS ENTRANTES CON EL MISMO USUARIO COMO DESTINO. 105
FIGURA 3.69: EL USUARIO SE ENCUENTRA OCUPADO CON UNA DE LAS LLAMADAS RETENIENDO EL RESTO. 105
FIGURA 3.70: EL USUARIO TIENE TODAS SUS LLAMADAS RETENIDAS. 106
FIGURA 3.71: INICIO DE UNA TRANSFERENCIA DE UNA DE LAS LLAMADAS RETENIDAS. 106
FIGURA 3.72: RESULTADO DE LA TRANSFERENCIA DE LA LLAMADA. 106
FIGURA 4.1: INTERFAZ DE ACCESO AL PRODUCTO IRI-421. 108
FIGURA 4.2: INTERFAZ DE INICIO DE IRI-421. 109
FIGURA 4.3: SELECCIÓN DE LA PESTAÑA DEL INTERFAZ DE OPERADOR. 110
FIGURA 4.4: VISTA GENERAL DEL INTERFAZ DE OPERADOR. 111
FIGURA 4.5: VISTA GENERAL DE LAS EXTENSIONES MONITORIZADAS. 112
FIGURA 4.6: EXTENSIÓN CON ESTADO NO DISPONIBLE. 112
FIGURA 4.7: EXTENSIÓN CON ESTADO LIBRE. 112
FIGURA 4.8: EXTENSIÓN CON ESTADO SONANDO. 112
FIGURA 4.9: ESTADO OCUPADO LLAMADA INTERNA. 113
FIGURA 4.10: ESTADO OCUPADO CON INTERLOCUTOR REGISTRADO. 113
FIGURA 4.11: ESTADO OCUPADO CON INTERLOCUTOR NO REGISTRADO. 113
FIGURA 4.12: ESTADO DEL USUARIO CON LLAMADAS RETENIDAS. 113
FIGURA 4.13: INICIO DE UNA LLAMADA INTERNA MEDIANTE EL TERMINAL. 114
FIGURA 4.14: VISTA GENERAL DE LA AGENDA. 114
FIGURA 4.15: VISTA DE EJEMPLO DE BÚSQUEDA DE UN PATRÓN EN LA AGENDA. 115
FIGURA 4.16: EJEMPLO DE LLAMADA ENTRANTE Y SALIENTE. 116
FIGURA 4.17: EJEMPLO DURACIÓN DE LA LLAMADA. 116
FIGURA 4.18: EJEMPLO DE PRIORIZACIÓN DE UNA LLAMADA ENTRANTE. 117

FIGURA 4.19: EJEMPLO SELECCIÓN USUARIO. 117
FIGURA 4.20: INICIO DE UN INTENTO DE TRANSFERENCIA DE UNA LLAMADA ENTRANTE A UN USUARIO DEL SISTEMA. 118
FIGURA 4.21: RESULTADO DE UN INTENTO DE TRANSFERENCIA DE UNA LLAMADA ENTRANTE A UN USUARIO DEL SISTEMA....118
FIGURA 4.22: LLAMADA ENTRANTE TRANSFERIDA A UN USUARIO DEL SISTEMA....119
FIGURA 4.23: INTENTO DE UNA TRANSFERENCIA DE UNA LLAMADA ENTRANTE A UN CONTACTO. 119
FIGURA 4.24: RESULTADO DE UNA TRANSFERENCIA DE UNA LLAMADA ENTRANTE A UN CONTACTO DE LA AGENDA....120
FIGURA 4.25: VISTA GENERAL DE LAS LÍNEAS DEL SISTEMA. 120
FIGURA 4.26: EJEMPLO DE OCUPACIÓN DE LOS CANALES. 121
FIGURA 4.27: EJEMPLO DE UN INTENTO DE ENCAMINAMIENTO DE UNA LLAMADA. 122
FIGURA 4.28: EJEMPLO DEL RESULTADO DE UN INTENTO DE ENCAMINAMIENTO. 122
FIGURA 4.29: ESPACIO RESERVADO PARA LA INTRODUCCIÓN DE DATOS. 123
FIGURA 4.30: EJEMPLO DE ENCAMINAMIENTO DE UNA LLAMADA POR UNA LÍNEA EN CONCRETO. 123
FIGURA 4.31: SELECCIÓN DE LA PESTAÑA DE CONFIGURACIÓN. 124
FIGURA 4.32: HERRAMIENTA DE SELECCIÓN DE EXTENSIONES. 124
FIGURA 4.33: VISTA GENERAL DEL INTERFAZ DE CONFIGURACIÓN. 125
FIGURA 4.34: RESULTADO DE LA ACTUALIZACIÓN DEL PANEL DE EXTENSIONES. 126
FIGURA 4.35: RESULTADO DEL USO DEL BOTÓN TRASLADAR. 127
FIGURA 4.36: EJEMPLO DE UN INTENTO DE CONFIGURACIÓN DE LA EXTENSIÓN DE UN USUARIO. 128
FIGURA 4.37: EJEMPLO DE UNA CONFIGURACIÓN MANUAL. 128
FIGURA 4.38: TABLA DE SELECCIÓN DE IDENTIFICADOR DE LAS EXTENSIONES. 128
FIGURA 4.39: EJEMPLO DE USO DEL LOGIN DE USUARIO COMO IDENTIFICADOR. 129
FIGURA 4.40: EJEMPLO DE USO DEL NOMBRE COMO IDENTIFICADOR DE USUARIO. 129
FIGURA 4.41: TABLA DE SELECCIÓN DE PANTALLAS. 130
FIGURA 4.42: VISTA DE LA CONFIGURACIÓN POR DEFECTO. 130
FIGURA 4.43: VISTA DE LA CONFIGURACIÓN PRIORIZANDO LAS EXTENSIONES EN LA VERTICAL. 131
FIGURA 4.44: VISTA DE LA CONFIGURACIÓN PRIORIZANDO LAS EXTENSIONES EN LA HORIZONTAL. 131
FIGURA 4.45: VISTA DE LOS BOTONES. 132
FIGURA 4.46: SELECCIÓN DE LA PESTAÑA DE AGENDA. 132
FIGURA 4.47: INTERFAZ DE VISUALIZACIÓN DE LA AGENDA DE USUARIO. 133
FIGURA 4.48: INTERFAZ DE CREACIÓN DE UNA NUEVA ENTRADA EN LA AGENDA DE USUARIO. 134

ÍNDICE DE TABLAS

TABLA 1: COSTE DEL SOFTWARE. 136
TABLA 2: COSTE DEL HARDWARE. 136
TABLA 3: RESUMEN DE LOS COSTES DE MATERIAL. 137
TABLA 4: RESUMEN DE COSTES DE LAS TAREAS DE UN INGENIERO SUPERIOR. 138
TABLA 5: RESUMEN DE COSTES DE LAS TAREAS DE UN INGENIERO TÉCNICO. 138
TABLA 6: RESUMEN DE COSTES DE LAS TAREAS DE UN ADMINISTRATIVO. 138
TABLA 7: RESUMEN COSTE DE EJECUCIÓN. 139

CAPÍTULO 1:
INTRODUCCIÓN Y OBJETIVOS

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Introducción.

El propósito del presente documento es realizar una descripción exhaustiva del diseño, desarrollo y la puesta en marcha del proyecto fin de carrera denominado como "Diseño e implementación de un terminal de control y monitorización para una centralita basada en *Asterisk*".

El trabajo realizado es una propuesta de herramienta de monitorización, de gestión y manipulación de llamadas en tiempo real de una centralita telefónica *PBX* (*Private Branch Exchange*) basada en *Asterisk*.

La herramienta cuyo desarrollo es el objeto de este proyecto fin de carrera, representa una pieza de un proyecto mayor de empresa. Dicho proyecto, que comprende al terminal desarrollado, consiste en el desarrollo software para realizar la configuración y la gestión vía web de una *PBX*.

1.2. Objetivos del Documento.

El objetivo principal de la presente memoria es conseguir una descripción detallada del trabajo realizado a lo largo de la puesta en marcha del proyecto, y a su vez, pretende dejar claro el modo de uso de la aplicación desarrollada, y por último, se considera importante que este documento pueda llegar a servir de guía para algún compañero que quiera continuar con el trabajo y mejorar el producto.

Ampliando el resumen del párrafo anterior, se pretende que la memoria contenga una descripción técnica detallada del esquema global del producto, profundizando en cada módulo por separado, así como los enfoques de diseño seguidos, las consideraciones tomadas y los distintos problemas encontrados y las soluciones que se consiguieron aplicar. De esta forma, se pretende conseguir que el presente documento sirva como guía para desarrolladores que tengan la intención de continuar con el trabajo aquí comenzado, o de desarrollar productos similares.

Para conseguir que también cumpla las características propias de un manual de usuario, se expondrá de forma detallada la forma de uso de la aplicación y las posibilidades de configuración que presenta ilustrándolos con ejemplos prácticos.

1.3. Estructura del Documento.

El presente documento dispone de una estructura de evolución lógica, es decir, primero se trata de realizar una introducción en el marco en el que se encuentra englobado el proyecto, que es el de la telefonía *IP* (*Internet Protocol*). A continuación se describe de forma más concreta el trabajo llevado a cabo, a la vez que se explican las funcionalidades y las prestaciones del producto desarrollado. Como siguiente paso, se realiza una estimación de los posibles costes del proyecto, y por último, se extraen conclusiones y se plantean posibles mejoras.

El documento está organizado en capítulos, a continuación se presenta una breve descripción de cada uno de ellos.

- En primer lugar se presenta el índice del documento.

- En este primer capítulo se presenta una descripción de los objetivos que se tuvieron presentes a la hora de realizar la memoria del proyecto fin de carrera. Una vez expuestos los objetivos, se presenta la organización en capítulos de la memoria y un pequeño resumen del contenido de cada uno de ello. A continuación se habla

brevemente sobre los objetivos del proyecto y se describen las fases en las que se ha dividido su desarrollo.

- El capítulo 2 pretende ser una introducción en el marco tecnológico en el que se engloba el proyecto desarrollado. Para tal fin, se presentará un breve estado del arte describiendo los inicios, la evolución y la situación actual de las tecnologías implicadas. A su vez se intentará realizar una comparación con tecnologías similares existentes en el mercado y justificar la razón de haber elegido las utilizadas finalmente.

- En el capítulo 3 se despliega la memoria técnica del proyecto, y se realiza un exhaustivo relato del trabajo de desarrollo realizado. Uno de los objetivos de este capítulo es tratar de justificar el diseño del producto y describir de forma detallada su estructura y cada módulo por separado. Para finalizar el capítulo, se explicarán las pruebas realizadas una vez justificado su propósito y se comentarán los resultados obtenidos.

- El capítulo 4 recoge un completo manual de usuario describiendo meticulosamente los pasos necesarios a seguir para conseguir el comportamiento deseado de la herramienta construida.

- En el capítulo 5 se presenta el presupuesto y la memoria económica del proyecto, haciendo una estimación del coste del desarrollo del producto software implementado teniendo en cuenta todas las etapas del proceso realizando una clasificación de todas las tareas efectuadas.

- En el capítulo 6 se pretende llevar a cabo un análisis del trabajo realizado, extraer las conclusiones pertinentes y discutir las posibles líneas de investigación futuras y mejoras necesarias a corto plazo, plasmando una valoración de los objetivos cumplidos y las posibles mejoras del producto.

- Por último, el capítulo 7 consiste en una completa bibliografía con las distintas fuentes de información utilizadas a lo largo del proyecto, así como algunas reseñas útiles si se desea ampliar los conocimientos sobre las tecnologías con las que se ha trabajado. A su vez, dicho capítulo adjunta un glosario de la colección de siglas utilizadas a lo largo de la presente memoria.

1.4. Fines del proyecto.

El proyecto fin de carrera cuya descripción se realiza en la presente memoria, pretende desarrollar una herramienta que permitiera la gestión en tiempo real de las llamadas realizadas en una centralita basada en *Asterisk*.

Dicha herramienta fue pensada para ser un instrumento más para mejorar la gestión de una centralita de *VoIP* (*Voice over Internet Protocol*). En concreto, fue ideada para facilitar el trabajo de las secretarias-telefonistas, ya que su propósito era permitir hacer más fácil su labor de distribución de las llamadas entrantes entre el resto de los miembros de la empresa.

Es importante destacar que esta herramienta es sólo una pieza de un proyecto más grande, que consiste en el desarrollo del software que permite la configuración y gestión vía web de una centralita basada en *Asterisk*. La totalidad del proyecto se denomina *IRI* (*Inteligencia de Red InTecDom*) y está siendo desarrollada por la empresa *InTecDom*. Dicho producto pretende abarcar toda la parte de la configuración de una centralita, desde la creación de usuarios del sistema, pasando por la generación del plan de llamadas hasta la configuración de las tarjetas *FXS* (*Foreign Exchange Station*) y *FXO* (*Foreign Exchange Office*) que se deseen añadir al sistema.

1.5. Fases del Proyecto.

En el caso de enfrentarse a la realización de un proyecto de cierta magnitud y relevancia, se hace indispensable una correcta planificación del trabajo. En esta sección se expone y explica la forma de organizar el trabajo a lo largo de todo el proyecto.

1.5.1. Fase I: Investigación previa y Estado del arte.

El propósito de esta primera etapa de la evolución del proyecto fue realizar una investigación exhaustiva de la situación actual del mercado para así asegurar la viabilidad del producto, su posible interés en el mercado, las características que debería cumplir para conseguir ser competitivo y finalmente decidir si existía un interés real para llevar la idea a cabo.

Una vez realizado dicho estudio inicial, se procedió a la recopilación de información sobre las tecnologías que se pretendían emplear en el transcurso del proyecto. En esta fase fue clave la elaboración del llamado "Estudio del arte" para conocer mejor las posibles alternativas y la evolución en el tiempo de las distintas tecnologías para a continuación, teniendo en cuenta las restricciones impuestas por la empresa en la que ha sido desarrollado el proyecto, proceder a la elección de las herramientas adecuadas.

1.5.2. Fase II: Estudio detallado del API de *Asterisk Manager.*

Una parte importante del proyecto se basa en la comunicación de un programa *Java* con el *Manager de Asterisk*. Para desarrollar dicha comunicación se empleó la librería denominada *Asteriskjava*, la cual consiste en un conjunto de clases que permite construir aplicaciones *Java* que interactúan con el servidor *PBX Asterisk*.

Para desarrollar dicha comunicación fue necesario familiarizarse con la colección de eventos y acciones que proporciona el *Asterisk Manager* y a continuación estudiar cuáles estaban contempladas en la versión de *Asteriskjava* utilizada.

1.5.3. Fase III: Estudio de una *PBX Asterisk*.

Esta tercera etapa consistió en el montaje de una maqueta de una centralita basada en *Asterisk* y realizar la configuración necesaria para disponer de un escenario válido.

La idea básica del proyecto consiste en determinar el estado concreto de la centralita gracias a la información proporcionada por el manager de *Asterisk*. Para ello era vital familiarizarse con los eventos generados en cada situación concreta, con su estructura y los datos que proporcionaban. Para conseguir dicho propósito, se procedió al montaje de la centralita, para lo cual, inicialmente simplemente se instaló una copia del software libre *Asterisk 1.4.24-RSP*. A continuación se realizó la configuración de un sencillo plan de llamadas que consistía en tener tres usuarios en el sistema que sólo podían comunicarse entre ellos. Para ello fue, lógicamente, también necesaria la configuración de los teléfonos de prueba de los que se disponía para la elaboración del proyecto. Una vez comprobado que el escenario básico, comprendiendo sólo llamadas internas, funcionaba correctamente, se procedió a completar el sistema añadiendo una línea de teléfono para observar el comportamiento de la centralita ante las llamadas externas, es decir, entrantes y salientes. Para ello se completó el plan de llamadas (la definición de un plan de llamadas se presenta en la sección 2.3) realizando la configuración pertinente y se volvió a comprobar que su actividad seguía siendo la correcta.

Los escenarios más complejos de configuración se realizaron en una etapa posterior del desarrollo del proyecto ya que el propósito de esta fase sólo era familiarizarse con los eventos generados por el *Asterisk* como consecuencia de la realización de las llamadas.

1.5.4. Fase IV: Diseño y desarrollo del software del producto.

Lo primero fue enfrentarse al diseño de un componente que funcionara de forma independiente del resto del programa y que permitiera capturar todos los eventos generados en el *Asterisk*. Como se comentará en el capítulo dedicado a la memoria técnica, se decidió emplear la tecnología de *jboss* para la ejecución de la aplicación. Una de las razones para decidirse emplear este servidor de aplicaciones fue la posibilidad que ofrece de una fácil construcción e implementación de programas que se ejecutan de forma independiente de los demás componentes del proyecto, permitiendo también una sencilla interacción entre ellos.

Una vez construido el servicio que sería el encargado de la comunicación con el *Asterisk Manager*, fue el momento de la planificación del módulo que se encargaría del análisis de la información obtenida y posteriormente, del envío del resultado a la página web para su visualización.

Tras la conclusión del diseño del producto se procedió a la implementación del código que proporciona las funcionalidades establecidas.

1.5.5. Fase V: Desarrollo de la interfaz del producto.

En una primera etapa del desarrollo del proyecto, se estuvo trabajando con una versión muy primitiva del interfaz, es decir, en los comienzos no se le dio prioridad al aspecto y funcionalidad de la página web sino que en principio se estuvo trabajando en el proceso de obtención de información dejando en un segundo plano su representación en pantalla.

Una vez se comprobó la correcta comunicación del servicio implementado y el Bean encargado del tratamiento de la información y el posterior envío de ésta a la página web, se procedió al diseño de la presentación.

Desde el primer momento se había decidido disponer de la pantalla con cuatro secciones bien diferenciadas ya que a la hora del diseño de la aplicación se identificaron cuatro bloques que resultaba interesante monitorizar. Estas cuatro categorías son:

- Las extensiones registradas en el sistema.
- Las llamadas en curso del sistema, tanto entrantes como salientes.
- La agenda del usuario encargado de la monitorización.
- Las líneas analógicas y digitales del sistema.

Partiendo de eso, se procedió al diseño de la página web propiamente dicho. Se escogieron colores representativos de la imagen de la empresa y se estudió su superposición de acuerdo con las reglas de accesibilidad. Finalmente, el aspecto final de la aplicación para la primera versión del producto fue la que se puede visualizar en la Figura 1.1.

Figura 1.1: Ejemplo de la visualización del Terminal de Operador.

1.5.6. Fase VI: Pruebas finales del producto.

Una vez obtenida la versión definitiva del software, se procedió a la definición de un banco de pruebas que permitiera comprobar que el producto obtenido satisfacía todas las exigencias impuestas y todos los objetivos definidos.

1.5.7. Fase VII: Puesta en marcha.

El objetivo de esta fase del proyecto residió en comprobar el funcionamiento de la herramienta desarrollada en un escenario real. Para observar el comportamiento del producto, se decidió realizar pruebas con la centralita de la empresa *InTecDom*. Dadas las características de las herramientas de desarrollo empleadas en el proyecto, no es necesario ejecutar el programa en la máquina que alberga físicamente la centralita de *Asterisk*. Es decir, para poder ver reflejada la actividad de la centralita deseada, era suficiente con lanzar el programa en local conectándose, a través de una red de área local, con el *Asterisk* y las bases de datos de la centralita real.

1.5.8. Fase VIII: Documentación del proyecto.

Como último paso de este proyecto se puede considerar el proceso de elaboración de la documentación. Se realizó una pequeña ficha con las características principales del producto, sus prestaciones y las exigencias de hardware que posee. A su vez, se elaboró un extenso manual de usuario con explicaciones detalladas del uso de la herramienta. Por último se confeccionó la presente memoria de proyecto fin de carrera.

CAPÍTULO 2:
ESTADO DEL ARTE

2. ESTADO DEL ARTE

2.1. Introducción.

El propósito de este capítulo es realizar un análisis del marco en el que se sitúa el presente proyecto. Para ello será necesario un repaso de los principales conceptos relacionados con la telefonía *IP* [1], las características principales de las tecnologías que se pretendían emplear en el desarrollo y la evolución en el tiempo de dichas tecnologías. También fue importante la realización de una comparativa con otras tecnologías presentes en el mercado y la justificación de las elecciones realizadas.

2.2. Descripción de *PSTN* y comparación con los servicios *VoIP*.

Las redes de telefonía públicas conmutadas, más conocidas por sus siglas en inglés *PSTN* (*Public Switched Telephone Network*) [2], han estado evolucionando desde que Alexander Graham Bell realizó la primera transmisión de voz sobre cable en 1876. Antes de entender cómo este servicio ha ido evolucionando hacia servicios de paquetes basados en *IP* [3], es importante analizar la evolución de este servicio a lo largo de los años. Esta sección se encuentra enfocada a explicar la evolución de las *PSTN* y la convergencia de esta tecnología con las de transmisión de datos.

2.2.1. El principio de las *PSTN*.

La primera transmisión realizada por Alexander Graham Bell se consiguió en 1876 mediante lo que se llama un circuito *ring-down*. Este tipo de circuitos se caracteriza por la inexistencia de marcación, básicamente una persona descolgaba el teléfono y otra persona se encontraba al otro lado.

A lo largo del tiempo, este simple esquema evolucionó de una transmisión unidireccional, mediante la cual sólo un usuario podía hablar, a una comunicación bidireccional en la que ambos usuarios podían comunicarse. Para poder transmitir la voz, el único hardware necesario era un micrófono de carbono, una batería, un electroimán y un diafragma de hierro.

También era necesaria la existencia de un cable físico entre cada localización a la que el usuario quisiera llamar. Para mostrar este concepto de forma gráfica se presenta la Figura 2.1 que muestra una red telefónica básica de cuatro usuarios en la que se puede contemplar como existía un cable que conectaba cada una de las localizaciones.

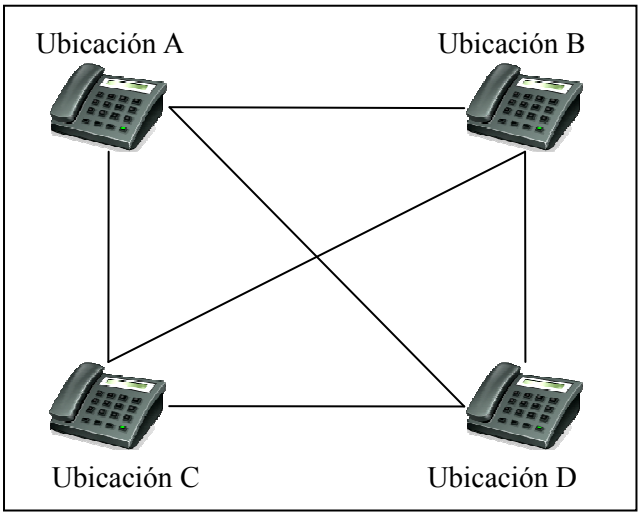

Figura 2.1: Ilustración de una red mallada.

Pronto se observó que instalar un cable entre cada localización no era efectivo a nivel de los costes. Este tipo de red se denomina mallada y es impracticable cuando el número de localizaciones es alto.

Debido a la imposibilidad de instalar un cable entre cada persona del mundo que quisiera acceder a un teléfono, se desarrolló un mecanismo capaz de enlazar diversas líneas. Mediante este mecanismo, llamado conmutador, sólo es necesario que cada usuario disponga de un cable a la oficina central de conmutación. En esta oficina se situaba un grupo de operadores que preguntaban al usuario a que número quería llamar y conectaban las líneas manualmente.

En nuestros días, pasados más de 100 años desde la invención del teléfono, las oficinas de conmutación basadas en operadores humanos han sido reemplazadas por conmutadores electrónicos capaces de servir miles de llamadas por segundo. En el siguiente apartado se explicarán las bases de la conmutación actual.

2.2.2. Entendiendo las modernas *PSTN*.

Aunque es difícil explicar cada componente de las *PSTN*, esta sección describe las partes más importantes de las *PSTN* actuales que permiten su funcionamiento. Se comenzará realizando una breve introducción y a continuación se describirán cada uno de los aspectos más importantes e imprescindibles para conocer las bases de la red telefónica pública conmutada.

2.2.2.1. Analógico vs Digital.

Todo lo que se escucha, incluyendo la voz humana es de naturaleza analógica. Hasta hace varias décadas, la red telefónica se basaba en infraestructura analógica. Aunque las comunicaciones analógicas son ideales para el ser humano, no son robustas ni eficientes a la hora de enfrentarse al ruido.

En las primeras *PSTN*, la transmisión analógica de la voz era tratada mediante amplificadores a lo largo de la línea, para aumentar el nivel de la señal. En la práctica, este mecanismo amplificaba el ruido, produciendo así un alto índice de ruido en la comunicación que podía ocasionar la inteligibilidad de la comunicación.

Si el usuario se encontraba lejos de la oficina de conmutación, la señal de la llamada debía ser amplificada. Este efecto era particularmente notorio en comunicaciones de larga distancia donde la señal y su ruido asociado pasaba varias etapas de amplificado.

Las redes modernas de telefonía están basadas en transmisión digital. En esta forma de transmisión, el ruido no es tan importante ya que los amplificadores utilizados no sólo aumentan la potencia de la señal sino que la reconstruyen, limpiando el ruido que pueda presentar. Este tipo de amplificador se llama repetidor.

De esta manera, en redes digitales, la señal de voz se mantiene libre de ruido. Una vez que las ventajas de la transmisión digital fueron patentes, la *PSTN* fue migrada a digital utilizando la codificación *PCM* (*Pulse Code Modulation*) [4].

2.2.2.2. PCM.

PCM es el método más común de codificar una señal analógica de voz en un flujo digital de unos y ceros.

Todas las técnicas de muestreo utilizan el teorema de *Nyquist* [5] que básicamente expresa que para recuperar una señal muestreada, el régimen de muestreo debe ser del doble de la frecuencia máxima de la señal.

El proceso de codificar una señal de voz en *PCM* es el siguiente:

- La señal de voz es tratada mediante un filtro paso bajo que elimina cualquier rastro de la señal que supere los 4 *KHz*. La voz es filtrada a esta frecuencia para reducir el nivel de diafonía en la red. Teniendo en cuenta el teorema de *Nyquist*, la señal se muestrea a 8000 muestras por segundo.

- Una vez que la señal es muestreada, se convierte a un código digital. Cada muestra es representada por un código que indica la amplitud de la onda y el instante en el que la muestra fue tomada. El código utilizado por *PCM* en las redes telefónicas asigna 8 *bits* para la amplitud y utiliza un sistema de compresión logarítmico para asignar más bits a las señales de menor amplitud. De esta manera el régimen binario de una señal telefónica codificada en *PCM* es de 64 *Kbps*.

2.2.2.3. Bucles Locales, Troncales y Comunicación entre los Conmutadores.

La infraestructura telefónica comienza con un simple par de cobre enlazando el hogar con la central de conmutación. Este cable se conoce como bucle local o bucle de abonado.

La línea de comunicación entre el hogar del usuario y la oficina de conmutación se llama línea telefónica. La línea de comunicación entre varias centrales de conmutación se llama línea troncal.

Al igual que no es eficiente conectar cada usuario con cada ubicación a la que quiera llamar, tampoco es eficiente conectar todas las centrales de comunicación de forma mallada. De esta manera, las centrales de conmutación se encuentran conectadas de forma jerárquica. Cada central de conmutación local se comunica mediante una línea troncal con una central de conmutación, que gestiona un conjunto de centrales de conmutación locales. Estas centrales de conmutación se conocen como centrales de conmutación de clase 4.

2.2.2.4. Servicios y Aplicaciones de la *PSTN*.

Casi en cualquier industria es normalmente más sencillo adquirir nuevos ingresos ampliando los servicios ofrecidos que adquirir nuevos usuarios. La *PSTN* no es diferente, los servicios ofrecidos por las operadoras han ido creciendo a lo largo de los años, ofertando un gran número de nuevos servicios y buscando el máximo de ingresos por cliente posible. Recientemente han aparecido numerosos servicios que no se encontraban disponibles hace unos años. Estos servicios pueden dividirse en servicios orientados al cliente y servicios *CLASS*.

Los servicios orientados al cliente se basan en nuevas características incluidas en la oficina local de conmutación, no en la infraestructura entera, y proveen información entre diferentes circuitos de comunicaciones. Los servicios *CLASS* requieren de señalización dentro de la red (*SS7* (*Signaling System Number 7*)) [6] y proveen de servicios extremo a extremo.

La siguiente lista presenta una serie de servicios populares que normalmente se pueden encontrar en las redes actuales.

- Llamada en espera. Notifica a los clientes que ya tienen establecida una llamada, de una llamada entrante.
- Desvío de llamada. Permite a los clientes el desvío de una llamada a otro número.
- Llamada a tres. Posibilita la multiconferencia de tres personas.

Con el despliegue de la señalización *SS7*, múltiples servicios avanzados de comunicaciones fueron desplegados. A continuación se presentan varios ejemplos de servicios *CLASS* que actualmente se pueden encontrar en las redes de telefonía.

- Identificación de llamada.
- Bloqueo de llamadas. Permite especificar una lista negra de números a los que no se les permite llamar a un número determinado.
- Bloqueo de identificador de llamada. Permite realizar llamadas “anónimas” que no muestran el identificador de llamada.
- Llamada automática. Permite que en el caso de que el destinatario se encuentre ocupado con otra llamada, la red automáticamente realice otro intento de comunicación cuando la línea se encuentre disponible.
- Rellamada.

- Llamadas de larga distancia.
- Tarjetas prepago.
- Números de Calling Party. Permite la tarificación extra o gratuita de distintos números, el beneficio de las llamadas a estos números se reparte entre el propietario de los números y la operadora.
- *VPN* (*Virtual Private Network*). Permite a una compañía el uso de números de marcación privados [7].
- Líneas privadas alquiladas. Permite alquilar líneas de datos y comunicaciones para uso exclusivo de la empresa.

2.2.3. Factores influyentes en la convergencia de las redes de voz y de datos.

Este apartado presenta los factores principales que fuerzan la integración entre las redes de datos y de voz.

2.2.3.1. Inconvenientes de la PSTN.

Aunque la *PSTN* es efectiva y realiza un buen trabajo para la aplicación para la que está diseñada, existe una serie de factores de negocio que están forzando a los operadores a cambiar hacia una nueva red en la que la voz es una aplicación más. Las razones para esta transformación son las siguientes:

- Los datos han superado a la voz como tráfico primario en muchas redes diseñadas para voz. El tráfico de datos ha aparecido en las redes de voz, copando las mismas. El tráfico de datos tiene unas características muy diferentes a las de la voz, como un uso de ancho de banda mayor y variable. En las redes actuales el tráfico de datos y de voz es diferenciado para que este último no afecte a la calidad de las llamadas.

- La *PSTN* no es capaz de crear e implantar nuevos servicios con la velocidad necesaria. Debido a la complejidad de las redes telefónicas, la creación e implementación de servicios es muy difícil. Debido a la desregularización del mercado de las telecomunicaciones, existe una gran competencia en el mercado. Las operadoras intentan mantener su base de clientes y para ello es necesario la creación de nuevos servicios que mantengan a los usuarios interesados por la oferta. La *PSTN* está basada en una infraestructura en la que únicamente los manufactureros de los equipos desarrollan nuevos servicios. Esto implica que los operadores dependen de uno, o como mucho de un par de distribuidores para incorporar nuevos servicios a la red. Es imprescindible una infraestructura más abierta que permita a diferentes empresas la creación de nuevos servicios para mantener el nivel de generación de nuevos servicios que los clientes demandan. Así mismo, no es posible con la infraestructura existente, la creación de nuevos servicios por compañías ajenas, ya que los equipos son completamente propietarios.

- La arquitectura creada para la voz no es lo suficientemente flexible para llevar datos. Además, debido a que la señalización de control y la lógica de servicios están fuertemente integrados, no es posible realizar pequeños ajustes en la red que mejorarían sus prestaciones para voz o datos.

- También es importante notar que la conmutación de circuitos requiere el establecimiento de un circuito permanente de 64 *Kbps* entre ambos terminales. Mientras la línea está ocupada, se transmitan datos o no, esta capacidad está reservada y nadie más puede utilizarla.

2.2.3.2. Coste económico.

Datos/Voz/Video no puede converger con la red desplegada *PSTN*. Los bucles locales existentes hoy en día no permiten la convergencia de esta serie de servicios debido a la falta de ancho de banda. Si estos servicios se quieren incorporar a la oferta de los operadores, son necesarias nuevas tecnologías como *ADSL* (*Asymmectric Digital Suscriber Line*) [8] o *RDSI* (*Red Digital de Servicios Integrados*) [9] para proveer el suficiente ancho de banda. La imposibilidad de convergencia implica un elevado coste económico correspondiente a la gestión y administración de dos redes separadas.

2.2.3.3. Características de redes IP.

Las razones expuestas anteriormente provocan la convergencia de las distintas comunicaciones de voz, vídeo y datos en una única red. Las nuevas redes creadas se denominaron redes *IP* y se basaron en la tecnología de conmutación de paquetes, la cual consigue usar la capacidad disponible de una forma mucho más eficiente que las redes basadas en la conmutación de circuitos.

La introducción de las presentes redes permitió el desarrollo de *VoIP*, que es el mecanismo que permite transportar conversaciones telefónicas en paquetes *IP*. Existen dos grupos de alternativas tecnológicas de *VoIP*, que básicamente son tecnologías propietarias y sistemas abiertos. Al primer grupo pertenecen productos como Skype [10], mientras que en el segundo se encuentran estándares abiertos basados en H.323 [11], *SIP* (*Session Initiation Protocol*) [12] o *IAX2* (*Inter-Asterisk eXchange*) [13], cuya breve descripción se presenta a continuación.

- H.323.

Éste fue el primer protocolo abierto de la telefonía *IP*. Forma parte de un conjunto de protocolos recogido bajo el nombre H.32X, que es una serie de estándares que definen los elementos necesarios para la telefonía a través de las redes *IP*.

El proceso de llamada bajo este protocolo es bastante complejo, la petición de servicio no es ortogonal y depende de los elementos que intervienen en ésta.

- *SIP*.

Como su nombre indica, se trata del protocolo que gestiona el inicio de sesión y permite aportar un conjunto de funciones para el procesamiento de las llamadas.

SIP es un protocolo de control de la capa de aplicación que puede establecer, modificar, y terminar sesiones multimedia como llamadas telefónicas por Internet. *SIP* puede también invitar participantes a sesiones existentes, como conferencias.

- *IAX2*.

IAX2 es un protocolo de señalización de telefonía *IP* que consigue utilizar un reducido número de bits en la cabecera y está diseñado para permitir la comunicación entre centralitas y clientes de *Asterisk*. Se trata de una alternativa al protocolo de señalización *SIP*, fue creado como parte del desarrollo de la *PBX Asterisk* [14] y resuelve muchos problemas y limitaciones tanto de *SIP* como de H.323.

Una característica importante de *IAX2* es que ha conseguido disminuir considerablemente el número de bits de la cabecera necesarios para el encaminamiento de los paquetes. Además, es capaz de agrupar los paquetes de distintas conversaciones dirigidas hacia la misma dirección en la red.

La principal ventaja que ofrecen las redes *IP* reside en que sí que permiten el desarrollo de nuevos servicios imposibles de crear en las redes tradicionales.

A las reducciones de costes mencionadas anteriormente, gracias a la implantación de una única red, se unen las debidas al uso de la telefonía *IP* que permite a las empresas rebajar el gasto en llamadas entre sus distintas delegaciones o sucursales.

Otra ventaja importante de las redes *IP* radica en la capacidad de utilizar el ancho de banda sólo cuando lo requiere, a diferencia de los circuitos dedicados característicos de la *PSTN*.

2.3. Introducción a *Asterisk*.

Asterisk es una aplicación de software libre que se distribuye bajo licencia *GPL* (*General Public License*) y que implementa una central telefónica (*PBX*). Como en el caso de cualquier *PBX*, puede conectar un número determinado de teléfonos para hacer llamadas entre sí e incluso conectar a un proveedor de *VoIP* o bien a una *RDSI*, tanto accesos básicos como primarios.

Mark Spencer, de *Digium*, fue el creador de *Asterisk* y actualmente es su principal desarrollador, junto con otros programadores que han contribuido a corregir errores y añadir novedades y funcionalidades.

Cuando desarrolló *Asterisk*, Mark Spencer, todavía era estudiante de ingeniería informática en la Universidad de Auburn, Alabama. En 1999 creó la empresa "*Linux Support Services*" con el objetivo de dar soporte a usuarios de Linux. Para ello necesitaba una centralita telefónica, pero ante la imposibilidad de adquirirla, dado sus elevados precios, decidió construir una con un *PC* (*Personal Computer*) bajo Linux, utilizando lenguaje C [15].

Posteriormente "*Linux Support Services*" se convertiría en "*Digium*" en el año 2002, redirigiendo sus objetivos al desarrollo y soporte de *Asterisk*.

Originalmente, *Asterisk* fue desarrollado para el sistema operativo *GNU/Linux* [16], en actualidad también se distribuye en versiones para los sistemas operativos *BSD* (*Berkeley Software Distribution*) [17], *MacOSX* [18], *Solaris* [19] y *Microsoft Windows* [20], aunque la plataforma nativa (*GNU/Linux*) es la mejor soportada de todas.

Asterisk incluye muchas características que antes de su creación sólo estaban disponibles en costosos sistemas propietarios *PBX*, como es el caso de buzón de voz, conferencias, *IVR* (*Interactive Voice Response*), distribución automática de llamadas, y muchas otras funcionalidades más. Dichas funcionalidades pueden seguir creciendo ya que los usuarios pueden crear nuevas escribiendo un plan de llamada (*dialplan*) en el lenguaje de script de *Asterisk* o añadiendo módulos escritos en lenguaje C o en cualquier otro lenguaje de programación soportado por *Linux*.

Para conectar teléfonos estándar analógicos son necesarias tarjetas electrónicas telefónicas *FXS* o *FXO* fabricadas por *Digium* [21] u otros proveedores, ya que para conectar el servidor a una línea externa no basta con un simple módem.

Una de las características más interesantes de *Asterisk* es que soporta muchos protocolos *VoIP* como pueden ser *SIP*, H.323, *IAX* y *MGCP* (*Media Gateway Control Protocol*) [22]. *Asterisk* puede interoperar con terminales *IP* actuando como un registrador y como pasarela entre ambos.

Desde hace algún tiempo, *Asterisk* ha sido adoptado en algunos entornos corporativos como una gran solución de bajo coste junto con *SER* (*SIP Express Router*).

Existe multitud de empresas relacionadas con *Asterisk*. La mayor parte de ellas siguiendo uno de los modelos de negocio más habituales del software libre, como es el de aportar valor añadido al software, en este caso mediante el diseño, instalación, formación y mantenimiento de centralitas telefónicas basadas en *Asterisk*.

Digium, la empresa creada por Mark Spencer, amplía este modelo de negocio tanto con la venta de hardware específico, fundamentalmente tarjetas de comunicación, como con la venta de software propietario, entre el que destaca el "*Asterisk Business Edition*", aplicación basada en *Asterisk* a la que se le incorporan ciertas funcionalidades.

Una versión estable de *Asterisk* está compuesta por los siguientes módulos:

- **Asterisk:** Ficheros base del proyecto.
- **Dahdi:** Soporte para hardware. Drivers de tarjetas. Este paquete antes era conocido como **Zaptel.**
- **Addons:** Complementos y añadidos del paquete *Asterisk*, es opcional.
- **Libpri:** Soporte para conexiones digitales, es opcional.
- **Sounds:** Aporta sonidos y frases en diferentes idiomas.

Cada módulo cuenta con una versión estable y una versión de desarrollo. La forma de identificar las versiones se realiza mediante la utilización de tres números separados por un punto. Teniendo desde el inicio como primer número el uno, el segundo número indica la versión, mientras que el tercero muestra la revisión. En las revisiones se llevan a cabo correcciones, pero no se incluyen nuevas funcionalidades.

En las versiones de desarrollo el tercer valor siempre es un cero, seguido de la palabra "beta" y un número, para indicar la revisión.

La última versión estable disponible para su libre descarga en el momento actual es la 1.6.2.

2.4. Tecnologías similares a *Asterisk.*

Desde que la versión 1.4 de *Asterisk* salió al mercado en 2006, este proyecto ha ganado inercia y una importante masa de usuarios. Con el creciente número de despliegues y base de usuarios, la estabilidad y escalabilidad aparecen como factores críticos para el futuro de esta plataforma. Aunque la empresa *Digium* se encuentra detrás de la plataforma *Asterisk*, mantener una plataforma de esta complejidad y conseguir una buena estabilidad de la misma, no es una tarea sencilla.

A pesar del esfuerzo y la dedicación de los desarrolladores de *Asterisk* para conseguir un sistema mejor, son todavía muchas las voces quienes piden cambios controvertidos.

Sin embargo, aunque como se ha comentado, sigue siendo una tecnología que no se puede considerar robusta y definitiva, en el momento actual, representa la solución más popular de una centralita de código abierto, aunque en los últimos años se haya proliferado la aparición de numerosas tecnologías similares. Aunque no proporcionan la misma funcionalidad y algunos ni siquiera están diseñados para ser una *PBX*, se presentan a continuación algunas de las tecnologías alternativas a *Asterisk*. En la Figura 2.2 se reproducen los logotipos de algunas de esas alternativas descritas en los apartados posteriores.

Figura 2.2: Alternativas a *Asterisk*.

2.4.1. FreeSWITCH.

FreeSWITCH [23] es un conmutador de llamadas software y de código abierto desarrollado por un grupo de desarrolladores de *Asterisk* que no estaban de acuerdo con una serie de decisiones sobre la arquitectura. Específicamente, estaban en desacuerdo con las siguientes características:

- Arquitectura monolítica, el procesamiento, el interfaz de usuario y los datos, residen en una misma entidad de red.
- Soporte limitado de distribuciones *NIX y de *GCC* (*GNU Compiler Collection*) [24].
- Soporte limitado de muestreo de voz a 8 *KHz*.

FreeSwitch se encuentra disponible desde 2006 y soporta un nutrido grupo de protocolos como *SIP*, *IAX2*, H.323, *XMPP* (*Extensible Messaging and Presence Protocol*) [25] y *Google Talk* (*Jingle*) [26] entre otros.

Las aplicaciones desarrolladas haciendo uso de la librería *FreeSwitch* pueden estar escritas en numerosos lenguajes de programación, encontrándose entre ellos C, C++, JavaScript, Perl, Python, etc., y empleando también diversos motores, estando disponible un *API* (*Application Programming Interface*) de *FreeSwitch* para cada uno de los casos.

La principal característica de *FreeSwitch*, que permite aumentar el rendimiento con respecto a la tecnología de *Asterisk*, es el hecho de que el primero emplea un hilo por cada canal, permitiendo con dicho diseño reducir considerablemente las situaciones de punto muerto y prácticamente eliminar los problemas de conocer el estado de los canales del sistema. Otra ventaja importante con respecto a *Asterisk* es que *FreeSwitch* emplea un algoritmo tipo *hash* para tratar los canales del sistema en lugar de una lista enlazada, como es el caso de *Asterisk*. Lo cual permite que mientras un hilo externo tenga una referencia a un cierto canal, éste no pueda ser eliminado o alterado por otro.

La presente tecnología, por el momento no está tan difundida como *Asterisk*, sin embargo, son muchos los que ya empiezan a considerarla como un posible sustito de éste, debido al hecho de que soluciona algunos de sus puntos más controvertidos.

2.4.2. SIP Express Router.

SIP Express Router [27] es considerado por muchos como una de las mejores soluciones de servidor de servicios *SIP* del mercado. Es un producto pensado para optimizar el rendimiento y la gestión de grandes volúmenes de usuarios *SIP*. *OpenSER* es el nombre original de la versión de código abierto de dicho sistema, que posteriormente pasó a llamarse *Kamailio* [28].

A diferencia de *Asterisk*, *OpenSER* no tiene prestaciones "*Media Gateway*", por lo que de por sí solo, no puede sustituir a una centralita avanzada, con lo cual no puede ser considerado equivalente a una centralita tradicional.

Sin embargo, está considerado como el servidor *SIP* más avanzado del mercado. Permite una gran escalabilidad y se trata de un entorno sumamente optimizado, basado en sistemas abiertos, y posibilitando la conexión de miles de usuarios de forma concurrente.

OpenSER ofrece un importante número de servicios, algunos de los cuales se enumeran a continuación:

- Redirector para llamadas *SIP* entrantes.
- Proxy.
- Servicio de registro de usuarios.
- Administración Web.
- Pasarela *SMS* (*Short Message Service*).
- Pasarela Jabber (*SIMPLE2Jabber*).
- Autenticación por Radius.
- Implementa *FCP* (*Firewall Communication Protocol*).
- Acceso web de los usuarios a sus registros.
- Registros por syslog.

Estas características hacen de *OpenSER* el sistema habitual para los proveedores de *VoIP* del mercado, o para grandes multinacionales. Al estar basado en sistemas abiertos, es habitual encontrar proveedores de *VoIP* que utilicen *OpenSER* como corazón del sistema, integrándolo con sistemas como *Asterisk* para interactuar con la red *PSTN* o efectuar labores de Contestador Automático.

Es muy común usar una plataforma mixta *OpenSER* + *Asterisk*, sobre todo cuando se está buscando funcionalidad de *PBX* y se tiene la necesidad de atravesar muchos tipos de redes. Se debe tener en cuenta que *SIP* no funciona correctamente con *NAT*s (*Network Address Translators*) [29] y el uso de un proxy puede aliviar ciertos problemas derivados de este hecho.

Asterisk puede ser configurado como un registrador de *SIP*, actuando como agente de usuario en el extremo final de la comunicación para luego crear una nueva llamada al destinatario real de la llamada, situándose de esta manera en el medio de la comunicación. Un proxy real, como *OpenSER*, nunca es el punto final de una llamada, maneja el control de llamadas en nombre de agentes de usuario, y no mantiene estado durante una llamada. *SER* soporta conexiones *SIP* con más funciones como la capacidad de registrar, funciones de servidor proxy, de redireccionar y es más escalable.

Kamailio fue ganador de "*InfoWorld's Best of Open Source Software Awards 2009*", en la categoría de mejor aplicación de red de código abierto.

2.4.3. sipX ECS.

sipX [30] es otra solución de una *PBX* basada en *SIP* que se puede considerar como uno de los mayores competidores de *Asterisk* a día de hoy. Su desarrollo comenzó en 1999 y en 2004 se convirtió en una fuente de código libre. Dicho producto está disponible tanto bajo una licencia *LGPL* (*Lesser General Public License*), como una solución comercial de la empresa *Pigtel*, que posteriormente fue adquirida por *Nortel*, denominada *SIPxchange*. *sipX ECS*

Enterprise Communication Server, representa un enfoque totalmente nuevo para hacer frente a la convergencia de voz y datos de las empresas.

Debido a las funcionalidades de proxy *SIP* que incorpora, *sipX* escala mejor que *Asterisk*. Siempre que la centralita sólo soporte usuarios *SIP* y que todos los flujos de voz sean entregados mediante *SIP*, este software es una buena opción frente a *Asterisk*. Pero es importante también destacar que *sipX* no soporta servicios de fax.

Tanto *sipX* como *Asterisk* tienen el mismo objetivo final: ofrecer una *PBX* basada en software, escalable, de bajo coste y que sea válida tanto para pequeñas como grandes instalaciones.

Ambos ofrecen más o menos las mismas funcionalidades pero diferenciándose en ciertas características importantes. *sipX* es menos vulnerable al *jitter* y a los retardos, y posee una interfaz web con una administración más sencilla. Pero *Asterisk* soporta más protocolos, por ejemplo H.323, *SCCP* (*Skinny Client Control Protocol*) [31], *MGCP*, *IAX*.

Por tanto, si lo que se desea es sólo un proxy *SIP*, *SIPxchange* podría ser más adecuado. Pero si se necesita hacer transcodificación y usar diferentes protocolos, en ese caso es más adecuado *Asterisk*.

La oferta comercial de este producto proviene de la empresa alemana *Nilando*. Su plataforma de software es *Linux* y *FreeBSD*.

sipX incluye muchas funciones tradicionales de una centralita privada (*PBX*) como buzón de voz, respuesta interactiva de voz de sistemas, multiconferencias, etc.

Los principales componentes del sistema *sipX* son un switch o router *SIP* en torno al cual está diseñado. En contraste con su principal competidor de código libre, *Asterisk PBX* y la mayoría de las ofertas comerciales que utilizan *SIP* como protocolo de transporte, *sipX* no juega el papel de un back-to-back agente a usuario. Este planteamiento dio lugar a una forma modular y altamente escalable. Todos los componentes principales de *sipX* se ejecutan como servidores y no necesariamente tienen que residir en una sola máquina.

sipX se distingue de la mayoría de las otras fuentes de código abierto *VoIP PBX* por varias características:

- Toda la señalización se maneja utilizando el protocolo *SIP* nativo (*SIP* vs gatewaying o algún otro protocolo de señalización, por ejemplo, como se hizo en la *PBX Asterisk*).
- Una vez que la llamada está configurada, los paquetes se envían directamente entre las variables involucradas. Esto permite que la mayoría de los componentes *sipX* sean independientes de los medios de comunicación y sus codificaciones. Por ejemplo, la transmisión de video basada en *SIP* puede realizarse sin aumentar la carga sobre el sistema.
- La arquitectura del sistema es cliente-servidor y no es monolítica, es decir, los elementos *sipX* (proxy, servidor de medios de comunicación, etc) se comunican entre sí a través del protocolo *SIP* y se puede ejecutar en diferentes terminales.
- El sistema administrativo es una interfaz basada en la web (frente a un interfaz de comandos de *Asterisk*) y su nombre es *sipXconfig*. *sipX* se adhiere a la filosofía de aplicación de *SIP*, con muchas funciones y con un importante apoyo en las variables como teléfonos, pasarelas o sistemas de voz y no en su totalidad en los componentes básicos como es un proxy. Esto mejora la escalabilidad, como ya se ha comentado, pero hace muchas características dependientes del apoyo a los parámetros del sistema telefónico.

2.4.3.1. Aplicaciones.

sipX ECS se dimensionó para el uso por pequeñas y grandes empresas que podrían llegar hasta unos 10000 usuarios. El mayor despliegue anunciado públicamente gestiona 5000 usuarios.

sipX está disponible en múltiples plataformas como *FreeBSD* y las principales distribuciones de Linux, incluyendo Red Hat Enterprise Linux, Fedora Core, CentOS, Debian y otros. Empezando con la versión 3.10, *sipX* tiene soporte nativo para PowerPC.

2.4.3.2. Hardware.

sipX soporta el uso de hardware *SIP* conectado directamente a *Ethernet*, pero no soporta la conexión directa a líneas de teléfono, aunque existen pasarelas apoyadas comercialmente por *sipX*.

Además de lo ya mencionado, el sistema *sipX ECS* se usa como referencia de aplicación *SIP* estándar. De hecho, esta aplicación se utiliza en eventos de prueba de interoperabilidad organizadas por *SIP Forum*. Un sistema automatizado de interoperabilidad *SIP* basado en *sipX* se ofrece de forma gratuita por *Pingtel* Corp [32]. Es utilizado principalmente por los fabricantes de teléfonos *SIP* para verificar que su producto cumple con la especificación *SIP* estándar.

2.4.3.3. sipX vs Asterisk.

A continuación se presenta una breve comparativa de las características del sistema *sipX* y *Asterisk* que podría ayudar a decidir cuál de las dos tecnologías es más adecuada para un caso concreto.

2.4.3.3.1. Estructura.

Asterisk: Esta tecnología tiene como objetivo servir como una *IP PBX* basada en las normas de protocolos de interconexión, tales como H.323, *PSTN*, *SCCP*, *MGCP*, *IAX* y *SIP*, que le permitan interconectarse con una variedad de sistemas diferentes. Aunque *Asterisk* admite todos los principales protocolos de señalización, no se trata de un servidor proxy *SIP* que proporciona un enrutamiento global de sesiones *SIP*.

sipX: Es una tecnología de infraestructura básica que se basa en *SIP* del *IETF* (*Internet Engineering Task Force*) [33] para conferencias multimedia sobre *IP*. Los flujos viajan directamente entre los puntos finales con sólo control de llamada, sin necesidad de de pasar por el servidor. Este enfoque reduce el volumen de trabajo necesario en el sistema *IP PBX*, dando ventaja arquitectónica al servidor *SIPxchange*.

2.4.3.3.2. Características.

Asterisk: *Asterisk* ofrece tanto características clásicas de una *PBX* como funciones más avanzadas. La *IP PBX* interopera con las normas tradicionales basadas en sistemas de telefonía y los sistemas de *VoIP*. *Asterisk* ofrece numerosas características avanzadas que se asocian generalmente con gama alta (y de alto coste) de propiedades de las *PBX*, como extensiones de itinerancia, llamada y música en espera, buzón de voz o marcado por nombre.

sipX: *sipX* tiene un conjunto de características paralelas a las de *Asterisk*, incluido las características sofisticadas y de alto nivel, que demuestra la valía de basar un sistema completamente en el estándar *SIP* que se puede utilizar para crear características de una *IP PBX*.

2.4.3.3.3. Calidad de voz.

Asterisk: La calidad de voz se ve obstaculizada en *Asterisk* por su estructura tradicional de *PBX*. Se desperdicia ancho de banda utilizando las líneas de transmisión para voz y datos de señalización en la *IP PBX*. Esto hace que el sistema sea vulnerable al *jitter* y los retardos.

sipX: A diferencia de *Asterisk*, *sipX ECS* no envía las comunicaciones a través del servidor. En lugar de ello, lo hace punto a punto mediante medios de información de enrutamiento, que se utiliza para proporcionar una mayor calidad de voz con una reducción de retardo y *jitter*.

2.4.3.3.4. Administración.

Asterisk: La fácil administración no es un sello distintivo de *Asterisk*, la principal herramienta de administración de Linux es una interfaz de línea de comandos. La configuración del sistema se maneja a través de ficheros de texto. Sin embargo, hay disponibles algunas herramientas de configuración, como *AMP* (*Asterisk Management Portal*).

sipX: El *sipX ECS* proporciona una interfaz Web para la administración del sistema que permite una rápida y fácil configuración y gestión del sistema. Los usuarios nuevos pueden ser aprovisionados y configurados de forma individual o como grupo. Debido a que *sipX ECS* está totalmente basado en *SIP*, agregar pasarelas y otros dispositivos *SIP* es tan fácil como añadir un usuario al sistema.

2.4.3.3.5. Apoyos.

Asterisk: La tecnología de *Digium* cuenta con el apoyo de una amplia selección de documentación y guías de solución de problemas, así como una comunidad de usuarios que ofrece foros de discusión, listas de correo y chats en tiempo real. También hay una gran cantidad de empresas que se especializan en el diseño de *Asterisk*, su despliegue y en servicios de mantenimiento. Se ofrece también un curso acelerado de *Asterisk* en *VoIP*-Noticias para realizar los primeros pasos.

sipX: *Pingtel* ofrece un gran apoyo a sus clientes con suscripciones activas a través del portal proporcionado. La compañía también ofrece un programa Jump Start que ofrece asistencia por teléfono y correo electrónico, ya sea en una primera instalación o en una actualización importante, ampliación o reestructuración del despliegue de *SIPxchange ECS*. Además, los clientes de *Pingtel* pueden adquirir varios paquetes de apoyo complementario para reforzar el apoyo incluido en su suscripción. Los clientes que no tienen una suscripción activa, y necesitan apoyo pueden obtener ayuda de *Pingtel* solicitándosela.

2.4.4. YATE.

Se trata de una alternativa más de una centralita de código abierto, las siglas que forman su nombre significan "*Yet Another Telephony Engine*" [34]. El presente proyecto está desarrollado bajo licencia *GPL* y está escrito en lenguaje C++. Dicho software ofrece las siguientes funcionalidades:

- Servidor *VoIP*.
- Cliente *VoIP*.
- Multiconferencia.
- Pasarela entre *VoIP* y *PSTN*.

- Servidor y/o cliente telefonía *IP*.
  - Proxy *SIP*.
  - Router *SIP*.
  - Servidor de registro *SIP*.
  - Servidor y/o cliente *IAX*.
  - Servidor o pasarela *MGCP*.
- *IVR*.
- Call center.

Entre las características de esta tecnología se encuentra su gran flexibilidad, ya que el diseño de paso de mensajes empleado permite crear aplicaciones antes impensables. Se trata de un sistema portable porque se puede utilizar en múltiples sistemas operativos y arquitecturas hardware. YATE supera a algunos productos de la competencia en términos de fiabilidad y escalabilidad, además se trata de un software muy estable ya que durante el proceso de desarrollo de dicho producto se llevan a cabo numerosas pruebas de fiabilidad.

2.4.5. Otras alternativas a Asterisk.

Otra alternativa es *OpenPBX*, este software es un desarrollo basado en *Asterisk* 1.2. Mantiene algunas de las características y aplicaciones de *Asterisk* 1.2 que fueron eliminadas de *Asterisk* 1.4 y ha mejorado significativamente el proceso de compilación frente a *Asterisk* 1.2. Uno de los mayores inconvenientes de esta plataforma comparada con *Asterisk*, es la falta de soporte en la comunidad. Este hecho fue especialmente visible cuando un fallo de seguridad de *SIP* requirió que todas las plataformas parchearan sus sistemas. *OpenPBX* fue una de las que más tardó en proveer una nueva versión del sistema que solucionara el problema.

Para concluir, se nombran a continuación algunas de las restantes alternativas a *Asterisk* existentes en el mercado pero con una menor repercusión que las descritas antes. Son por ejemplo: Bayonne [35], CallWeaver [36], openSIPS [37], Trixbox [38], etc. También existen numerosas soluciones que se basan en *Asterisk*, como por ejemplo: VerCom [39], Abraxas, AskoziaPBX, softTelecom, etc.

A parte de las alternativas capaces de reemplazar *Asterisk* en mayor o menor medida, existen ciertas características de *Asterisk* que han sido identificadas como fuentes de problemas y complejidad. Estas partes o subfunciones son las siguientes:

- *Asterisk IP PBX (MeetMe)*: A pesar de la riqueza de características de *Asterisk*, a la hora de utilizarlo en muchas aplicaciones específicas ha sido necesario rediseñarlo para hacer frente a las demandas de los usuarios y al deseo de la comunidad. *MeetMe* es bien conocido por superar en eficiencia a mecanismos de conferencia tradicionales, basados en *TDM* (*Time-Division Multiplexing*) [40]. Debido a la naturaleza de *VoIP*, el número de participantes en la conferencia se encuentra limitado por la *CPU* (*Central Processing Unit*), memoria del sistema y ancho de banda, en lugar de por el número de puertos telefónicos, como en los sistemas tradicionales. Sin embargo, *MeetMe* se basa en dispositivos *Zaptel* (fabricante de dispositivos electrónicos de comunicaciones, entre otros *Calling Cards*) como fuente de temporización. De hecho, al analizar más profundamente esta dependencia, se puede observar que el driver *Zaptel* es el encargado de mezclar el audio de diversas fuentes en los buffers, tanto provenientes de líneas analógicas como digitales. En ausencia de una tarjeta *Zaptel* encargada de esta función, *MeetMe* utiliza un driver software que emula el funcionamiento de la tarjeta a bajo nivel, utilizando el reloj de alta resolución del Kernel. Esta dependencia ha sido identificada como fuente de numerosos problemas al aumentar considerablemente la complejidad de las operaciones.

- *Voicemail:* Una de las características más elogiadas de *Asterisk*, es la capacidad de enviar correos de voz como adjuntos en emails. Sin embargo, estos mismos usuarios pronto descubren que los correos de voz borrados del buzón de correo, no son eliminados del buzón de voz. Esto se debe a que no existe una manera más o menos sencilla de interactuar entre el gestor de correo y el gestor de buzón de voz. Existen algunas soluciones a este problema, por ejemplo, se ha implementado un chequeo periódico que comprueba los mensajes en el buzón de voz y los eliminados del buzón de correo. Otras posibles soluciones pasan por realizar modificaciones a los servidores de correo *IMAP* (*Internet Message Access Protocol*) [41] para que sean capaces de interactuar con los buzones de voz de *Asterisk*. La última novedad en este servicio es la creación de un servidor *IMAP* que gestiona mensajes de voz, permitiendo de esta manera al usuario la gestión de mensajes de voz y correo electrónico de manera homogénea e independiente del mecanismo de acceso, ya sea mediante el teléfono o mediante un cliente de correo electrónico.

- *Fax y Asterisk*: La aplicación o extensión a *Asterisk* más utilizada para lidiar con faxes es *HylaFAX* [42]. *HylaFAX* es una máquina de fax software que es capaz de enviar y recibir faxes y que utiliza la librería *SpanDSP* (*Span Digital Signal Processing*) [43] para realizar las tareas de procesado digital. Todo esto se incorpora a *Asterisk* por medio de dos aplicaciones, "*app_rxfax*" y "*app_txfax*". *Asterisk* recibe el fax como una imagen *TIFF* (*Tagged Image File Format*), la cual puede ser convertida a *PDF* (*Portable Document Format*) y enviada por email a cualquier usuario con un sencillo programa definido en el *dialplan*. El mayor inconveniente de estas aplicaciones es que a pesar de ser muy eficiente, son poco estables, funcionando raramente y no se distribuyen con *Asterisk*, siendo necesario su reinstalación cuando se realiza cualquier actualización de *Asterisk*.

2.5. Descripción de *Asterisk*.

Asterisk es un software completo que implementa una *PBX*, está desarrollado en Linux y que provee todas las configuraciones que se esperan de una *PBX* y más. *Asterisk* proporciona servicios *VoIP* y puede interoperar con equipos de telefonía estándar básicos usando un hardware relativamente de bajo coste.

Asterisk admite una amplia gama de protocolos *TDM* para el manejo y transmisión de interfaces de telefonía tradicional. *Asterisk* soporta al tipo de señalización estándar americano y europeo en asuntos de sistemas de telefonía, permitiendo ser un nexo entre las redes integradas de datos y voz de siguiente generación y la infraestructura existente. *Asterisk* no sólo soporta los equipos de telefonía tradicionales sino que también los habilita con capacidades adicionales. Esta centralita software permite la conectividad entre las redes *PSTN* y las redes *IP* en tiempo real.

2.5.1. Módulos Cargables.

Asterisk presenta un núcleo principal encargado de gestionar las funciones principales de la centralita. Para llevar a cabo dichas funciones, el núcleo de *Asterisk* se apoya en módulos que le proporcionan la abstracción de los codecs, las interfaces y los protocolos que se estén empleando. Esto le permite a *Asterisk* utilizar cualquier hardware conveniente y tecnología disponible, ahora o en el futuro para realizar sus funciones esenciales, conectando hardware y aplicaciones.

Dichos módulos son conocidos como *API*s y destacan los siguientes cuatro de ellos.

- API Canal: Este *API* maneja el tipo de conexión que el cliente está usando, ya sea una conexión *VoIP*, *RDSI* o algún otro tipo de tecnología. Se cargan módulos dinámicos para manejar los detalles de la capa más baja de estas conexiones.

Un canal es una conexión que conduce una llamada entrante o saliente al sistema *Asterisk*, dicha conexión puede realizarse con redes de telefonía tradicional, tanto analógica como digital o con redes *IP*. *Asterisk* soporta distintos tipos de canales entre los cuales destacan los siguientes:

- *SIP*, *IAX2*: Canales que implementan el comportamiento descrito en los protocolos de *VoIP*.
- *Zap*: Canales encargados de gestionar las líneas analógicas y digitales.
- *mISND*: Canales encargados de gestionar las líneas *RDSI*.

- API Aplicación: Se trata de un conjunto de módulos que son capaces de proporcionar a la centralita diversos servicios de valor añadido.
  Las aplicaciones son un conjunto de comandos cuyo destino es ser ejecutados por *Asterisk*. Algunos ejemplos de aplicaciones más sencillas son:
  - *Hangup*: Permite colgar una llamada.
  - *Dial*: Supone el inicio de una llamada.
  - *Monitor*: Pone en marcha la grabación de una llamada.
  - *PlayBack*: Realiza la reproducción de un fichero de sonido.

- API Traductor del Codec: Su función reside en la carga de diferentes formatos de codecs utilizados para comprimir y codificar la señal de voz. Un codec es un algoritmo compresor/decompresor, es decir, un conjunto de transformaciones utilizadas para digitalizar la voz. Los codecs convierten tanto la voz en datos como los datos en voz. Existen muchas formas de digitalizar audio y cada una de esas formas resulta en un tipo de codec. En general, se puede asumir que a mayor compresión se va a obtener mayor distorsión. Un codec se considera mejor que otro cuando es capaz de ofrecer mejor calidad de voz usando la misma cantidad de ancho de banda.

- API Formato de archivo: Se encarga de la lectura y escritura en varios formatos de archivos para permitir el almacenamiento de datos en el sistema, como puede ser la grabación de las conversaciones.

Usando estos *API*s *Asterisk* alcanza una completa abstracción entre sus funciones básicas como un servidor de sistema *PBX* y la variedad tecnológica existente, o en desarrollo, en el área de la telefonía.

La estructura modular, es lo que le permite a *Asterisk* integrar hardware de telefonía implementado y la tecnología de paquetes de voz emergentes hoy en día.

La aplicación *API* provee el uso flexible de aplicaciones modulares para realizar cualquier acción demandada, también permite un desarrollo abierto de nuevas aplicaciones para satisfacer necesidades o situaciones únicas.

En conclusión, la filosofía modular permite un sistema flexible, permitiéndole al administrador diseñar la mejor y más satisfactoria trayectoria para los usuarios en el sistema *PBX* y también modificar la trayectoria de llamadas para satisfacer las cambiantes necesidades de la comunicación que nos concierne. En la Figura 2.3 se representa de forma gráfica la estructura de *Asterisk*, sus cuatro *API*s principales y las relaciones existentes entre ellas.

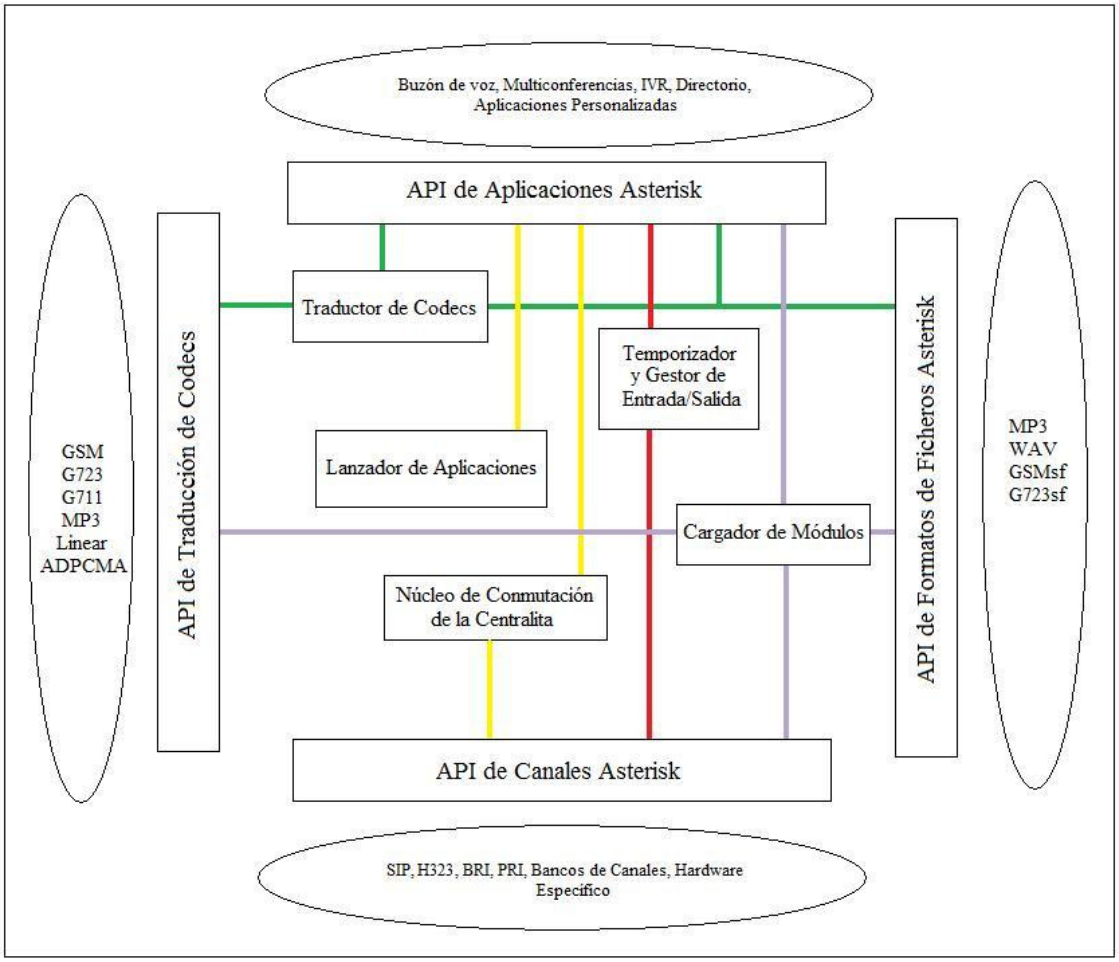

Figura 2.3: Estructura de *Asterisk*.

2.5.2. Funciones de *Asterisk*.

Como ya se ha comentado, *Asterisk* es una implementación de software libre de una centralita y es capaz de proporcionar todas las funcionalidades de las centralitas propietarias y además ofrece algunas aplicaciones y servicios no disponibles en las centralitas de pago.

Asterisk incorpora todas las funcionalidades de una centralita tradicional:

- Conexión con líneas de telefonía tradicional, mediante interfaces tipo analógico (*FXO*) para líneas de teléfono fijo o bien móvil y *RDSI* (*BRI* (*Basic Rate Interface*) o *PRI* (*Primary Rate Interface*)).
- Soporte de extensiones analógicas, bien para terminales telefónicos analógicos, terminales *DECT* (*Digital Enhanced Cordless Telecommunications*) [44] o bien equipos de fax.
- Soporte de líneas *IP*: *SIP*, H323 o *IAX*.
- Soporte de extensiones *IP*: *SIP*, *SCCP*, *MGCP*, H323 o *IAX*.
- Música en espera basada en archivos *MP3* (*MPEG-1 Audio Layer 3MPEG-1 Audio Layer 3*) y similar.
- Autoaprovisionamiento automático de los teléfonos.
- Autenticación.
- Roaming.
- Integración con bases de datos.

- Identificación de número llamante.
- Fax.
- Funciones básicas de usuario:

    - Transferencia de llamada, tanto ciega como con previa consulta.
    - Desvíos de llamada.
    - Capturas de llamadas.
    - Aparcamiento de llamadas.
    - Llamada directa a extensión.
    - Retrollamada.

A su vez, *Asterisk* ofrece funcionalidades avanzadas cuya implantación supondría un coste muy elevado en sistemas tradicionales, algunas de ellas son:

- Buzón de voz: Sistema de contestador automático personalizado por el usuario.
- *IVR*: Sistema automatizado de respuesta que permite redirigir las llamadas entrantes en función de las opciones seleccionadas el emisor de la llamada.
- *CDR*: Las siglas son de "Call Detail Record" y se trata de un informe detallado de las llamadas tanto recibidas como realizadas. Toda la información pertinente se almacena en la base de datos y puede emplearse, por ejemplo, para realizar la tarificación o simplemente para el establecimiento de las estadísticas de llamadas.
- *ACD*: (Automatic Call Distribution) Sistema Automático de Distribución de Llamadas Entrantes, más conocido como Call Center. Consiste en el establecimiento de una cola hacia la cual se dirigen todas las llamadas entrantes, las cuales se distribuyen entre los agentes designados a tal efecto, según algún criterio preestablecido.
- Sistema de Audioconferencias: Sistema que permite la conexión remota de diferentes usuarios que quieren mantener una reunión virtual y suministra la correcta gestión y control de los usuarios que se incorporan a ella.
- Integraciones con reconocimiento de voz.

2.5.3. Estructura de Asterisk.

El software de *Asterisk* se encuentra organizado en una serie de directorios, algunos de los cuales contienen diversa información complementaria para la ejecución del programa, mientras que otros, albergan ficheros de configuración cuya edición determinará el funcionamiento de la centralita.

Los mencionados directorios se instalan por defecto en ubicaciones predeterminadas, sin embargo, dichas ubicaciones también son configurables mediante la edición del archivo denominado "*asterisk.conf*".

A continuación se comentan los directorios más relevantes.

- **/var/lib/asterisk**

Este directorio contiene diferentes librerías que emplea *Asterisk*, que son unos subprogramas necesarios para el funcionamiento del núcleo de *Asterisk*. Alberga, por ejemplo, un directorio con sonidos que se utilizan para las diferentes aplicaciones, o directorio que contiene las claves *RSA* (*Rivest, Shamir and Adleman*) para la autenticación de llamadas con el protocolo *IAX2*. También contiene un directorio que almacena los archivos de audio que emplearán las aplicaciones que usan *music on hold*.

- **/usr/lib/asterisk/modules**

En este directorio se almacenan los módulos de *Asterisk* que han sido compilados. Dependiendo de la versión de *Asterisk*, se cargan todos los módulos disponibles o sólo los que se van a utilizar realmente. También existe la posibilidad de escoger los módulos según las necesidades mediante la edición del archivo de configuración denominado "*modules.conf*". Sin

embargo hay que tener cuidado ya que existen dependencias entre los distintos componentes, provocando inconsistencias la ausencia de algunos de ellos.

- **/var/spool/asterisk**

Se trata de un directorio que almacena archivos generados por *Asterisk* durante su funcionamiento. Dichos archivos pueden ser los referentes a la actividad del fax, pueden ser mensajes de audio grabados por el contestador automático, las grabaciones de las conferencias llevadas a cabo o similar.

- **/var/log/asterisk**

En el presente directorio se almacenan los log o archivos de eventos generados como producto de la actividad de la centralita. Un archivo log es un archivo de texto plano en el que se refleja toda acción realizada por un programa, servidor o aplicación. El grado de detalle de la información almacenada, al igual que otros parámetros, también es configurable mediante la manipulación del archivo *logger.conf*.

- **/etc/asterisk**

Este directorio se encarga de albergar todos los archivos de configuración de *Asterisk*, a excepción de *zaptel.conf* que se ubica directamente en el directorio /etc/ porque cualquier otro software debe tener acceso al driver de *zaptel*.

Como ya se ha comentado, este directorio contiene los archivos que permiten realizar la configuración de los codecs empleados, de los logs, del *CDR* y otros muchos aspectos de la centralita. Sin embargo, el archivo de configuración que podría considerarse como el más importante es el llamado *extensions.conf* ya que es el archivo donde se define el plan de marcado, conocido como *dialplan*.

Existe una variedad de *Asterisk* denominado *Realtime*, dicha variedad permite almacenar la configuración de la centralita en una base de datos en lugar de emplear los archivos de configuración dispuestos en el directorio /etc/asterisk. La ventaja que presenta esta opción reside en el hecho de que de esta forma, *Asterisk* puede leer la configuración realizada en tiempo real, es decir, el administrador puede realizar modificaciones que serán incorporadas al comportamiento de la centralita sin la necesidad ni de reiniciar el *Asterisk* ni de volver a cargar los módulos.

2.5.4. *Dialplan*.

En el plan de numeración o de llamadas, denominado con la palabra inglesa *dialplan*, se almacenan todas las acciones y situaciones que la *PBX* debe ser capaz de manejar.

El *dialplan* es verdaderamente el corazón de cualquier sistema *Asterisk*, ya que define cómo se ocupa *Asterisk* de la entrada y la salida de las llamadas. En resumen, se compone de una lista de instrucciones o pasos que *Asterisk* deberá seguir. A diferencia de los sistemas telefónicos tradicionales, el sistema de *dialplan* de *Asterisk* es completamente personalizable.

El *dialplan* se compone de cuatro partes principales: los contextos, las extensiones, las prioridades y las aplicaciones y en los siguientes apartados se realiza una descripción de cada una de ellas.

Tal y como se ha descrito en el apartado anterior, el plan de marcado está almacenado en el archivo de texto denominado *extensions.conf*. En este archivo, a las extensiones se les atribuyen ciertas acciones, cada extensión pertenece a un contexto, ya sea contexto por defecto o un contexto específico que se haya creado, como llamadas *SIP* entrantes, llamadas de salida de larga distancia *PSTN*, llamadas locales, llamadas entre oficinas o alguna otra cosa. Todos los usuarios conectados a *Asterisk* pertenecen a un contexto, especificado en el canal de archivo de configuración, que es donde *Asterisk* busca la forma de cómo manejar las llamadas para cada usuario.

A continuación se describen los componentes del plan de marcado de *Asterisk*.

2.5.4.1. Contextos.

Los *dialplan* se dividen en secciones llamadas contextos. Los contextos describen grupos de extensiones. En pocas palabras, se mantienen diferentes partes del *dialplan* separadas y se describe la interacción de unos con otros. Una extensión que se define en un contexto es completamente independiente de otras definidas en cualquier otro contexto, a menos que la interacción sea específicamente permitida. Los contextos se caracterizan por su nombre, que se indica entre corchetes.

Todas las instrucciones situadas después de la definición de un contexto forman parte de ese contexto, hasta el siguiente contexto definido. Al comienzo del *dialplan*, hay dos contextos especiales llamados "*general*" y "*globals*".

El contexto representa uno de los parámetros principales a la hora de crear un canal, que supone la base del funcionamiento de la centralita.

Uno de los propósitos más importantes de los contextos es hacer cumplir la seguridad. Si no se cuida el diseño del *dialplan*, se puede permitir involuntariamente el uso fraudulento del sistema a personas no autorizadas.

2.5.4.2. Extensiones.

Dentro de cada contexto se definen una o más extensiones. Una extensión es una instrucción que *Asterisk* va a seguir, la cual se desencadena por una llamada entrante o el marcado de unos dígitos determinados. Una extensión puede ser de tipo literal, que son las normales que pueda definir un administrador de la centralita con conocimientos limitados. Las extensiones de tipo estándar, que son las que ya están definidas por defecto. Y por último, se pueden definir extensiones de tipo especial, que son extensiones más elaboradas y necesitan un estudio más profundo del sistema.

La sintaxis para la definición de una extensión es la palabra exten seguida de una flecha formada por los signos igual y mayor, de la siguiente manera:

$$exten \Rightarrow$$

A continuación se pone el nombre de la extensión. Cuando se trata de sistemas de telefonía, se tiende a pensar en las extensiones como en los números que permiten llamar de un teléfono a otros, sin embargo en *Asterisk* tiene muchas más funcionalidades.

Una extensión completa se compone de tres elementos:

- El nombre o número de la extensión.
- La prioridad que le corresponde. Cada extensión puede incluir varios pasos, el número del paso se denomina prioridad.
- La aplicación o comando que lleva a cabo alguna acción.

Estos tres componentes están separados por comas, de la siguiente manera:

$$exten \Rightarrow nombre, prioridad, aplicación()$$

2.5.4.3. Prioridades.

Cada extensión puede tener varias etapas, llamadas prioridades. Cada prioridad se numera secuencialmente, comenzando con 1.

En el ejemplo presentado se especifica una extensión que define primero responder al teléfono, lo cual se refleja con la prioridad número 1 y, a continuación, colgar, y por tanto se le asigna la prioridad 2.

$$exten \Rightarrow 1000, 1, Answer()$$

$$exten \Rightarrow 1000,\ 2,\ Hangup()$$

Es importante asegurarse de que las prioridades empiecen en 1 y se numeren consecutivamente ya que si se salta una prioridad, *Asterisk* no continuará.

Sin embargo, para la definición de extensiones con un número elevado de prioridades, a partir de la versión 1.2 de *Asterisk*, se introdujo el concepto de la prioridad *n*. La ventaja que presenta consiste en que cada vez que *Asterisk* se encuentra con esa prioridad, se encarga de coger el número de prioridad anterior y le añade una unidad. Esta funcionalidad permite realizar modificaciones en el *dialplan* de una forma más fácil y cómoda.

2.5.4.4. Aplicaciones.

Las aplicaciones son una parte muy importante del *dialplan*. Cada aplicación realiza unas acciones en relación al canal actual, tales como la reproducción de un sonido, la aceptación de un tono de entrada, o colgar la llamada.

Algunas aplicaciones, como la de *Answer()* y *Hangup()*, no necesitan otras instrucciones para llevar a cabo su función. Otras aplicaciones requieren una información adicional. Este tipo de información, llamada argumentos, se coloca después del nombre de la extensión, entre paréntesis y separados por comas.

A continuación se enumeran algunos ejemplos de las aplicaciones:

- Conectar una llamada a un *voicemail* si el usuario no responde ni la primera ni la segunda llamada durante los primeros 20 segundos.
- Conectar una llamada a una multiconferencia.
- Transferir una llamada a otra *PBX Asterisk*.
- Bloquear llamadas de un llamante no deseado o no identificado.
- Crear llamadas en espera y dejar a grupos de agentes que manejen las llamadas entrantes.

2.5.5. *Asterisk* manager API.

Como se ha comentado en el capítulo introductorio, el trabajo llevado a cabo y que se documenta en la presente memoria, formaba parte de un proyecto mayor. Debido a lo cual, y aunque *Asterisk* represente la base principal del producto, la configuración y la puesta en marcha de la centralita no es muy relevante para la descripción de la labor realizada. Se ha considerado por tanto imprescindible dedicar un apartado especial a la descripción específica de la parte de *Asterisk* que ha sido utilizada para la construcción de la herramienta ideada para realizar la monitorización del sistema por el usuario operador.

En los siguientes apartados se realiza una descripción del llamado *Asterisk Manager* que es el componente de *Asterisk* más destacado en el proyecto descrito en esta memoria.

El *Asterisk Manager* permite a un programa cliente conectarse a *Asterisk* y emitir órdenes o leer eventos de un flujo *TCP* (*Transmission Control Protocol*) */IP* de la *PBX*. Es decir, estableciendo una sesión tras la autenticación, el programa cliente dispondrá de la posibilidad de enviar comandos o generar acciones que serán encaminados hacia *Asterisk*. Éste, a su vez, formará respuestas pertinentes o reaccionará generando eventos que describan la situación alcanzada.

Para poder permitir las conexiones descritas es necesario editar el fichero *manager.conf* donde se indica a quién se le permite dicho acceso y cuáles son el usuario y la clave.

En la versión de *Asterisk* 1.4 se incorpora *AJAM* (*Asyncronous Javascript Asterisk Manager*) [45] , que es un nuevo JavaScript [46] basado en la tecnología que permite a los navegadores web u otras aplicaciones *HTTP* (*HyperText Transfer Protocol*) [47] y páginas web acceder directamente al *Asterisk Manager Interface* (*AMI*) a través de *HTTP*.

2.5.5.1. Características del protocolo.

El protocolo tiene las siguientes características:

- Antes de emitir órdenes para *Asterisk*, se debe establecer una sesión de *Asterisk* manager.
- Los paquetes pueden ser transmitidos a cualquier dirección en cualquier momento después de la autenticación.
- La primera línea de un paquete tendrá una clave de "Acción" cuando se envía desde el cliente, pero "Evento" o "Respuesta" cuando es enviada de *Asterisk* al cliente.
- El orden de las líneas dentro de un paquete no es relevante, por lo que se puede utilizar cualquier lenguaje de programación nativo que ordene de manera eficiente los paquetes.
- *CRLF* (*Carriage Return LineFeed*) se utiliza para delimitar cada línea y una línea en blanco (dos *CRLF* en una fila) indica el final del comando y que *Asterisk* se queda a la espera de un nuevo proceso.

2.5.5.2. Tipos de paquetes.

El tipo de un paquete está determinado por la existencia de una de las siguientes claves:

- **Action:** Un paquete enviado por el cliente conectados a *Asterisk*, solicitando una acción particular que se debe realizar. Hay un finito, pero extensible conjunto de acciones a disposición del cliente, determinado por los módulos cargados actualmente en el motor de *Asterisk*. Sólo una única acción puede ser solicitada a la vez. El paquete de acción contiene el nombre de la operación a realizar, así como todos los parámetros necesarios.

- **Response**: La respuesta enviada por *Asterisk* a la última acción remitida por el cliente. Este paquete puede contener simplemente un mensaje de éxito o fallo o en cambio, contener una estructura completa con datos relevantes sobre la consulta realizada por el cliente.

- **Event**: los datos relativos a un evento generado desde dentro de *Asterisk* básico o un módulo de extensión.

Generalmente el cliente envía paquetes de Action para el servidor de *Asterisk*, el servidor *Asterisk* tramita la operación solicitada y devuelve el resultado en un paquete de respuesta, que a menudo sólo consiste en un mensaje que anuncia el éxito o el fracaso. Como no hay ninguna garantía en cuanto al orden de los paquetes de respuesta al cliente, por lo general incluye un ActionID, parámetro identificativo de cada paquete de Action que se envía de vuelta desde *Asterisk* en la respuesta correspondiente. De esta manera el cliente puede relacionar fácilmente Acción y Respuesta, de manera que se puede realizar el envío de paquetes de acciones de cualquier tipo sin tener que esperar a que lleguen los paquetes de respuesta pendientes antes de enviar la siguiente acción.

Los paquetes *Event* se utilizan en dos contextos diferentes: Por un lado, informar a los clientes sobre los cambios en el estado de *Asterisk*, como por ejemplo nuevos canales que se están creando o deshabilitando, agentes conectándose o desconectándose y por otro lado, son utilizados para el transporte de la carga útil de la respuesta para las acciones. Cuando un cliente envía una petición de generación de acción, *Asterisk* envía una respuesta indicando éxito o fracaso y que contiene el texto "Response: ... ". A continuación, se envía cero o más eventos que contienen la carga útil real y, por último, una acción completa el evento indicando que todos los datos ya han sido enviados. Los acontecimientos generados en respuesta a un evento de generación de acción y la acción, se completan con un evento que contiene la *ActionID* de la Acción de los paquetes que los ha provocado, para que pueda coincidir con ellos del mismo

modo que los paquetes de respuesta. Un ejemplo de un evento de generación de acción es la acción de *Status* que desencadena el envío de información sobre el estado para cada canal activo. Cuando todos los eventos de estado se han enviado, se genera un evento de *StatusComplete*.

2.5.5.3. *Asterisk* Manager Actions.

Se presentan a continuación algunos ejemplos de este tipo de paquetes empleados por la centralita, con una breve explicación de su funcionalidad.

- **AbsoluteTimeoutAction:** El AbsoluteTimeoutAction fija el tiempo máximo absoluto permitido para una llamada a un determinado canal. Hay que tener en cuenta que el tiempo se establece a partir de la hora actual hacia delante, sin contar el número de segundos de la llamada que ya ha sido iniciada. Esta acción corresponde al comando AbsoluteTimeout utilizado en el *dialplan*.

- **LoginAction:** Esta acción se encarga de autenticar la conexión. Haber conseguido establecer con éxito la conexión es la condición previa para el envío de cualquier otra acción, salvo para el caso de ChallengeAction. En caso de fallo de autenticación, se recibe un mensaje de error.

- **OriginateAction:** La OriginateAction genera una llamada saliente en el contexto dado, con la prioridad dada o para una determinada aplicación con opción de incluir parámetros. Si se quiere conectar a una extensión hay que utilizar las propiedades de contexto, la extensión y la prioridad.

- **ParkAction:** La presente acción permite realizar el aparcado de una llamada en concreto. Es decir, si se desea dar prioridad a una llamada, será necesario poner en espera al resto de las llamadas afectadas.

- **RedirectAction:** Esta acción provoca la sustitución de un canal por otro, lo que resulta en la realización de una transferencia de la llamada afectada.

- **HangupAction:** Su función reside en colgar el canal que recibe como parámetro.

Esto es tan sólo una pequeña representación de todas las acciones de las que dispone *Asterisk* que son las necesarias para proporcionar toda la funcionalidad existente hasta el momento.

2.5.5.4. Asterisk Manager Events.

A continuación se presenta una muestra de los eventos que se emplearon de los que dispone el *Asterisk Manager*.

- **DialEvent:** Cuando se está produciendo una llamada no relacionada con ninguna cola definida, sino que se hace directamente a una extensión, se produce un evento de este tipo. Proporciona información sobre tanto el número llamado como el llamante.

- **HangupEvent:** Tras la finalización de todo tipo de llamada, *Asterisk* genera un evento de este tipo para informar de lo ocurrido. Dicho evento proporciona información datos sobre la llamada, como el identificador único, el canal que se ha empleado, la identificación del que llamaba y una explicación de la causa por la cual esa llamada se ha finalizado. Dicho evento se genera para los dos canales implicados en una conversación.

- **HoldEvent:** En el momento en el que un operario pone en espera una llamada para realizar una transferencia o cualquier otra acción, se produce este tipo de evento. Permite saber el identificador único de la llamada y el canal empleado para llevarla a cabo.

- **LinkEvent:** Este evento tiene lugar cuando los dos canales, el primero asignado al llamante y el segundo al llamado, se unen para poder llevar a cabo la comunicación deseada. Proporciona información sobre los dos canales implicados y el identificador único asignado a ambos procesos.

- **ManagerEvent:** Se trata de una clase abstracta que sirve como estructura para todos los eventos que puedan ser tratados por *Asterisk*. Dicha estructura contiene los datos relativos a un evento generado desde dentro del *Asterisk* básico o un módulo de extensión. Hay una subclase concreta de **ManagerEvent** por cada evento que genera *Asterisk*.

- **NewCallerIdEvent:** Este evento se produce cuando el identificador del llamante asignado a un determinado canal cambia. Proporciona información sobre el identificador único de la llamada, el identificador del llamante, y el canal involucrado.

- **NewChannelEvent:** Cuando se crea un nuevo canal, sea cual sea su fin, *Asterisk* genera este tipo de evento para información de que existe un nuevo canal en el sistema.

- **NewExtenEvent:** Este evento es similar al anterior ya que se genera cada vez que se crea una nueva extensión en el sistema. Informa sobre el identificador único de la llamada, el canal, la extensión y una explicación de cuál es el fin de introducir esta nueva extensión.

- **NewStateEvent:** Este evento tiene lugar siempre que la comunicación cambia de estado. Se obtiene información sobre el canal implicado, el llamante y lógicamente el nuevo estado adquirido por el proceso.

- **PeerStatusEvent:** Este evento se genera periódicamente para informar sobre el estado del registro de los terminales configurados en el *Asterisk*.

- **RenameEvent:** El presente tipo de evento tiene lugar cuando se produce una reasignación de canales en el sistema. Es decir, cuando debido a la actividad de la centralita, un canal activo hasta el momento, es sustituido por uno nuevo. Por ejemplo, cuando se produce una transferencia de llamada, en el último paso, el canal del usuario que realiza la transferencia se sustituye por el correspondiente al nuevo interlocutor, pasando al estado inactivo el primero.

- **UnholdEvent:** Este evento se produce cuando la llamada anteriormente retenida para ser traspasada o para alguna otra acción, abandona dicho estado y la comunicación vuelve al canal temporalmente congelado.

- **UnlinkEvent**: En el momento en el se termina una conversación, se produce dicho evento para informar de que los canales que se habían unido al principio de ésta para que pudiera tener lugar, han dejado de estar en contacto. Proporciona información de los dos canales implicados en la situación descrita.

Una vez realizada la introducción a la tecnología principal utilizada para el desarrollo del proyecto que describe la presente memoria, el siguiente paso consiste en la exposición detallada de la parte técnica del trabajo llevado a cabo.

CAPÍTULO 3:
MEMORIA TÉCNICA

3. MEMORIA TÉCNICA

3.1. Introducción.

El propósito general de la realización del proyecto descrito en esta memoria era conseguir la implementación de una aplicación que permitiera la monitorización de una centralita basada en *Asterisk*. La idea del desarrollo de la presente herramienta surgió como parte de un proyecto mayor, que consiste en la interfaz de configuración y gestión de una inteligencia de red basada en *Asterisk*.

El estudio de mercado realizado por la empresa *InTecDom* [48] reveló que aunque ya existían productos que pretendían desplegar la misma idea, las implementaciones eran limitadas en funcionalidad. Se descubrió a su vez, que era posible cubrir una franja de mercado interesada en una producto completo, es decir, una inteligencia de red con todas las herramientas necesarias para su administración, configuración, gestión y monitorización completas. Tras dicho descubrimiento, la empresa se embarcó en la tarea de complementar su producto con diversas funcionalidades entre las que se incorporó la monitorización.

La intención era conseguir un producto plenamente integrado en la totalidad del proyecto del desarrollo de una inteligencia de red completa y que cumpliera todas las características exigidas.

Con el propósito de alcanzar el fin de desarrollar una aplicación que fuera capaz de proporcionar al usuario todas las herramientas necesarias para facilitar su trabajo de monitorización y a su vez permitir su interacción con la centralita, se fueron marcando diversos objetivos a lo largo de todo el trabajo realizado, cada cual correspondiente a una etapa del desarrollo determinada.

En una de las etapas iniciales, se estableció un objetivo clave cuya resolución repercutía directamente en la viabilidad del proyecto. Dicho propósito consistía en realizar un completo estudio de mercado para establecer sus necesidades reales y averiguar así, si el producto podía llegar a ofrecer algo nuevo y si tenía posibilidades de diferenciarse para conseguir encajar en el panorama existente. Se estudiaron los productos ya existentes en aquel momento y se concluyó que aunque sí que existían herramientas para monitorizar la actividad de una centralita, las funcionalidades que ofrecían eran limitadas y no llegaban a satisfacer las necesidades de los potenciales clientes. Las conclusiones sacadas del estudio realizado confirmaron las posibilidades del producto en el mercado y por tanto se procedió con su desarrollo.

Tras la etapa descrita, el siguiente gran objetivo que se marcó fue estudiar detalladamente el comportamiento del manager de *Asterisk* a lo largo de la actividad de la centralita para llegar a ser capaces de analizar de la forma más eficiente la información que éste puede llegar a ofrecer. Para ello hubo que realizar numerosas pruebas de actividad y observar detenidamente la respuesta que generaba el manager. Se realizó un estudio detallado de las estructuras de los eventos generados, de las secuencias de generación y de las respuestas de la centralita ante una intervención externa. Gracias a lo cual se consiguió obtener una visión detallada de las posibilidades que ofrecía el manager de *Asterisk*.

Una vez estudiado el comportamiento de la centralita se marcó el objetivo de estudiar en profundidad las posibilidades que ofrecía la librería de java que se decidió emplear para llevar a cabo la interacción con el manager de *Asterisk*. Para ello se estudió la estructura de la librería, el *API* disponible y de nuevo se realizaron pruebas para comprobar la respuesta real de la centralita ante el empleo de la herramienta seleccionada. Dicho estudio permitió establecer que la librería ofrecía las utilidades necesarias para llevar a cabo la implementación de las funcionalidades que se había propuesto que tuviera el producto desarrollado.

El siguiente objetivo importante marcado fue la creación de la aplicación diseñada que permitiera la gestión y monitorización de la centralita, y a su vez que el usuario encargado de la labor descrita dispusiera de la posibilidad de intervenir en la actividad de la centralita cuando él lo creyera conveniente. Para ello se realizó una clara definición de las funcionalidades que se

deseaba que ofreciera la herramienta y se procedió con su implementación haciendo uso de toda la información recopilada en las etapas anteriores.

El último objetivo primordial que se marcó a lo largo del desarrollo del producto descrito fue la validación de la solución implementada. En el momento de disponer de una primera versión considerada como funcional, se realizaron diversas pruebas para observar la respuesta de la herramienta ante la actividad de la centralita. La realización de dichas pruebas sirvió para localizar los posibles errores y comportamientos no deseados. Tras la delimitación de las incorrecciones, se procedió a su solución y al establecimiento de un protocolo de pruebas lo más eficaz posible para evitar la no detección de un comportamiento no deseado.

Una vez solucionados los problemas encontrados, la batería de pruebas diseñada permitió corroborar las funcionalidades de la herramienta realizada y concluir que respondía a los objetivos impuestos al principio del desarrollo.

3.2. Herramientas utilizadas.

Para conseguir el desarrollo del producto final que fuera capaz de responder a todas las exigencias impuestas inicialmente durante el proceso de diseño, se tuvo que recurrir a numerosas herramientas a lo largo del proceso.

En este apartado se procederá a la enumeración de las herramientas tanto software como hardware empleadas en la realización del proyecto, describiendo su utilidad y justificando su elección.

3.2.1. Herramientas software.

Para llevar a cabo el diseño y la posterior implementación de este proyecto, han sido necesarias distintas herramientas software, algunas de las cuales se detallan a continuación:

- Sistema Operativo Linux Debian.

Se escogió el mencionado sistema operativo debido a las ventajas que presentaba frente a cualquier otro disponible. La principal ventaja reside en el hecho de que *Asterisk* corre sobre Linux y por tanto, realizar el desarrollo del código en el mismo entorno facilitaba el trabajo de estudio e interacción con la centralita virtual. Otra ventaja importante residía en que el entorno de programación escogido para llevar a cabo el proyecto, tenía prestaciones visiblemente mejores en Linux.

- Asterisk.

La descripción detallada de la presente herramienta se puede encontrar en el capítulo de estudio de arte, en este apartado tan sólo se presentan las características principales de la versión del software empleada para la realización del proyecto.

La versión de *Asterisk* empleada para la ejecución del presente proyecto fue *1.4.24-RSP*. Sin embargo, fue necesaria la introducción de algunas modificaciones en el código fuente de dicha versión del software para conseguir el correcto funcionamiento de la centralita y las funcionalidades necesarias.

- Entorno de programación.

Una vez que se tiene clara la estructura de la aplicación que se desea llevar a cabo, es importante escoger el entorno de desarrollo más adecuado.

En el mercado existe multitud de entornos de desarrollo, más conocidos como *IDE*s (*Integrated Development Environment*), para trabajar en java. La elección del *IDE* adecuado

depende de muchos factores, como puede ser el coste del producto, la tendencia de mercado o el enfoque de la empresa desarrolladora. A continuación se realiza una breve presentación de los entornos de programación más populares en el panorama actual.

- **Bluej:** Es un *IDE* diseñado específicamente para la enseñanza a un nivel introductorio de la programación orientada a objetos. Posee las funcionalidades básicas de editar, compilar y depurar. Visualiza el código del proyecto en *UML* (*Unified Modeling Language*), mostrando las clases con las relaciones de herencia y dependencia entre ellas.

- **JCreator:** Es un *IDE* que consume muy pocos recursos con lo que puede ejecutarse en máquinas poco potentes, esto se consigue prescindiendo de las capacidades de otros *IDE*s, por ejemplo, no dispone de interfaz visual para el desarrollo de aplicaciones gráficas. No obstante, a diferencia de BlueJ este producto si es un *IDE* que puede ser empleado por un programador profesional.

- **Netbeans:** En esta ocasión ya se trata de un producto completo que requiere máquinas más potentes. Este *IDE* permite el diseño de aplicaciones de modo visual a través de su herramienta Matisse, siendo a partir de la versión 5.5 uno de los mejores editores para el diseño gráfico con Java según lo comentado en Internet. Además se trata de una herramienta gratuita.

- **JDeveloper:** Es la propuesta de Oracle de entorno integrado para lenguajes como Java, *HTML* (*HyperText Markup Language*), Javascript, *SQL* (*Structured Query Language*), *PL/SQL* (*Procedural Language/Structured Query Language*), *XML* (*eXtensible Markup Language*), *PHP*, *Oracle ADF* (*Application Development Framework*), *UML* y otros. Entre sus principales características se encuentra su fácil manejo, muy recomendable para desarrolladores poco experimentados. El *IDE* cuenta con características que cubren todo el ciclo de vida de una aplicación desde análisis basado en *UML*, desarrollo, debug, adicionalmente se puede hacer desarrollo orientado a base de datos, desarrollo para *XML*, desarrollo web (javaScript, *HTML*, *JSF* (*Java Server Faces*)). Además, este entorno cuenta con un servidor *J2EE* (*Java 2 Platform, Enterprise Edition*) embebido, lo cual facilita toda la parte de pruebas y despliegues.

- **Eclipse:** En la actualidad, éste es el *IDE* por excelencia dentro del mundo de desarrollo de Java. Su gran popularidad se debe a que cuando nació no había buenos entornos gratuitos de Java que le hiciesen competencia, pero a día de hoy Netbeans está empezando a gozar de cierta autoridad. Esta herramienta en sí no es sólo un editor Java sino que es un *IDE* extensible a base de plugins. Existen plugins que permiten configurar este entorno para prácticamente cualquier posibilidad desde hacer que Eclipse sea un editor, una herramienta de acceso a bases de datos o incluso un visor de mapas. Esta capacidad de adaptar el entorno con plugins presenta una dificultad añadida para el desarrollador Java novel ya que hay que aprender qué plugins se necesitan y dónde conseguirlos.

Para realizar el desarrollo del software se escogió Eclipse debido a que es una infraestructura abierta, diseñada para construir ambientes integrados de desarrollo que pueden ser usados para crear aplicaciones tan diversas como sitios web, programas java, programas C++, *EJB* (*Enterprise Java Beans*) [49] , servicios web, etc. Concretamente se utilizó Eclipse 3.4, la versión comercialmente conocida como *"Ganymede"* que era la versión más reciente y completa en el momento del inicio del proyecto.

En la última etapa del desarrollo, se realizó la actualización a la siguiente versión, la conocida bajo el nombre de *"Galileo"*, pero la mayor parte del desarrollo se realizó con la versión anterior de la herramienta.

- JDK 1.5.

"Java Development Kit" es el paquete que contiene el entorno de desarrollo de Java de Sun [50]. Es una herramienta que sirve para desarrollar programas Java y proporciona además el entorno de ejecución necesario.

- Jboss.

Jboss es un servidor de aplicaciones *J2EE* de código abierto implementado en Java.

La versión empleada en el desarrollo del proyecto fue 4.2.2.GA y en este apartado se realizará una breve descripción del dicho servidor de aplicaciones.

Jboss es el primer servidor de aplicaciones de código abierto que posee la certificación de compatibilidad con el estándar *J2EE* 1.4. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, *Jboss* puede ser descargado, utilizado y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular para desarrolladores, vendedores independientes de software y también, para grandes empresas. *Jboss* permite crear, implementar, integrar, organizar y presentar aplicaciones y servicios web en una arquitectura orientada a servicios.

Las características destacadas de *Jboss* incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.
- Soporte completo para *JMX* (*Java Management eXtensions*).

Jboss es un *EJB* Container, y por tanto se suele utilizar en conjunto con un Web Container, el cual en principio podría ser cualquiera aunque *Jboss* proporciona ya el *Tomcat* integrado, que es un Web Container y al encontrarse integrado la configuración necesaria será más sencilla.

Una vez instalado se crea la correspondiente estructura de directorios, de la cual, se procede a mencionar los más importantes.

- *bin*

Este directorio contiene los ejecutables utilizados por *Jboss*, principalmente los que permiten lanzar y parar el servidor de aplicaciones.

- *client*

Contiene todos los archivos o archivos de biblioteca necesarios para la parte del cliente que quiera comunicarse con el servidor *Jboss*.

- *lib*

Este directorio contiene las librerías empleadas por *Jboss* requeridos en cualquier modalidad.

- *server*

Este directorio contiene tres sub-directorios nombrados: *all*, *default* y *minimal*. Cada uno de los nombrados sub-directorios contiene los distintos archivos de configuración necesarios para ejecutar *Jboss* en diferentes modalidades.

La modalidad *all* permite la ejecución de *Jboss* para emplearse como "*Cluster*", ejecución de "*Web-Services"* y otras funcionalidades más. La modalidad denominada *default* incluye la configuración para ejecutar *Jboss* de manera básica, mientras el directorio *minimal* contiene los valores de configuración necesarios para ejecutar *Jboss* con requerimientos mínimos. El script de arranque proporcionado con *Jboss* emplea los valores del directorio

default, para emplear otra modalidad es necesario modificar dicho script de arranque que se denomina "*run.sh*" y como ya se ha comentado se alberga en el directorio *bin*.

Las aplicaciones que se deseen que se publiquen en el servidor deben ser colocadas en el directorio *deploy* de la modalidad de trabajo elegida.

- Doxygen.

Es una herramienta de código abierto que permite generar documentación y referencias de programación de paquetes de software de forma automática. Mediante el empleo de esta herramienta se ha realizado una documentación completa del código desarrollado. Se empleó un plugin denominado "*eclox*" que permitía usar la herramienta directamente desde el entorno de programación empleado, es decir *Eclipse*.

- Script.aculo.us.

Es una librería *JavaScript* que permite el uso de controles *AJAX* (*Asynchronous Javascript And Xml*), "*drag and drop*", que permite arrastrar y soltar elementos, realizar efectos de animación y otros efectos visuales en una página web. Se ha empleado dicha librería para la realización de la interfaz web de la herramienta.

- Mozilla Firefox.

La aplicación desarrollada está preparada para funcionar con este navegador en concreto, ya que forma parte del proyecto completo de Inteligencia de Red que ofrece también un terminal web para facilitar la configuración, cuyo desarrollo re realizó con dicho navegador. Las razones para escoger Mozilla Firefox entre todos los navegadores existentes se debe principalmente a las exigencias del potencial cliente que prefiere utilizar un navegador que sea multiplataforma, pudiéndose ser utilizado en Windows, Linux o Mac, lo cual reduce las posibilidades a Explorer y Mozilla Firefox. Se eligió finalmente Mozilla Firefox debido a sus diversas características entre las cuales hay que destacar:

  - Cumple los estándares.
  - Es multiplataforma y multilingüe.
  - Mayor estabilidad y rapidez.
  - Seguridad y privacidad.

- Asteriskjava.

Es una librería de clases Java que permite la fácil construcción de aplicaciones que interactúen con servidores *PBX Asterisk*. Mediante el uso de dicha librería se consiguió construir un servicio que fuera capaz de proporcionar en todo momento una información actualizada sobre el estado de la centralita.

La presente librería da soporte a dos interfaces que provee *Asterisk* y que son:

- ✓ Protocolo FastAGI (*Fast Asterisk Gateway Interface*): proporciona soporte a la implementación de todos los comandos disponibles para *Asterisk*.

- ✓ AMI (*Asterisk Manager Inteface*): proporciona soporte a la implementación del envío de acciones a la *PBX* y de la recepción de eventos de *Asterisk*.

Para la realización de este proyecto se ha empleado sólo la comunicación con la centralita mediante *AMI* ya que proporciona todas las herramientas necesarias para conseguir el propósito de la aplicación desarrollada. Sin embargo no se descarta el uso del resto de las

funcionalidades disponibles en las etapas del desarrollo de las posibles mejoras de la aplicación, ya que *AGI* (*Asterisk Gateway Interface*) proporciona las herramientas para controlar y alterar el plan de llamadas que podría resultar interesante para futuros desarrollos.

La funcionalidad del establecimiento de la comunicación con el Manager se consigue mediante la implementación de unas clases java reunidas en un paquete denominado manager, que se encuentra formado por los siguientes paquetes:

- *action*: Colección de clases java que permiten la implementación de las acciones que pueden ser enviadas a *Asterisk* mediante el interfaz del manager.
- *event*: Colección de clases java que representan los eventos generados por *Asterisk* y que pueden ser capturados por el interfaz del manager.
- *internal*: Conjunto de clases java cuya implementación es necesaria para proporcionar la funcionalidad total del interfaz del manager de *Asterisk*.
- *response*: Conjunto de clases java que representan las posibles respuestas generadas por *Asterisk* en respuesta a una acción enviada.

- JainSIP.

Se trata de otra librería java, que en esta ocasión proporciona la definición de las clases y métodos necesarios para conseguir la implementación de la pila *SIP*. Se hace uso de dicha librería para la construcción del servicio encargado de permanecer a la escucha de los eventos producidos por la centralita *Asterisk*.

- Sipp.

Se trata de una herramienta de software libre que permite generar tráfico *SIP*. Dicha herramienta se empleó en la fase final del proyecto para la realización de pruebas de estrés ya que Sipp es capaz de simular la generación de un gran número de llamadas sin necesidad de disponer físicamente de los canales necesarios para su tramitación. Gracias a la opción de generar tráfico *RTP* (*Real time Transport Protocol*) [51] en ambos sentidos de la que dispone, se consigue simular llamadas reales.

- Arquitectura J2EE.

Para la elaboración de la aplicación del Terminal de Operador se empleó la arquitectura *J2EE*, cuya breve descripción se presenta a continuación.

En la actualidad, a la hora de crear una aplicación de cierta envergadura, los desarrolladores recurren a modelos de capas. Dicho hecho se debe a que el modelo tradicional de dos únicas capas, es decir, modelo de cliente-servidor se ha quedado anticuado y ha evolucionado hacia estructuras más complejas para conseguir el desarrollo de aplicaciones distribuidas que trabajen de forma transaccional, manteniendo la rapidez, seguridad y escalabilidad.

J2EE es una plataforma basada en estándares creada para el desarrollo, despliegue y gestión de aplicaciones empresariales distribuidas con una estructura multicapa. Probablemente sea la plataforma más popular en el momento actual.

J2EE simplifica el desarrollo de este tipo de aplicaciones basándolas en componentes modulares y estandarizados y proporcionando un completo conjunto de especificaciones que aseguran la portabilidad de aplicaciones entre un amplio número de productos comerciales y de código abierto existentes, capaces de soportar *J2EE*.

Por otro lado, el concepto de código abierto promueve los beneficios del desarrollo corporativo y significa que los programas construidos según esta definición deben incluir el código fuente y permitir la distribución tanto en forma de fuente como compilado, sin coste alguno.

La plataforma está basada en la idea componente-contenedor. Los componentes son las aplicaciones, servicios, etc. Estos componentes se despliegan en contenedores, los cuales se ocupan de aspectos como la escalabilidad, la transaccionalidad, la concurrencia, la seguridad, etc.

Como se puede ver, un concepto clave de la arquitectura es el de contenedor, que dicho de forma genérica no es más que un entorno de ejecución estandarizado que ofrece unos servicios por medio de componentes. Los componentes externos al contenedor tienen una forma estándar de acceder a los servicios de dicho contenedor, con independencia del fabricante.

J2EE también proporciona la portabilidad de código, pudiendo desplegar nuestra aplicación en cualquier servidor, sobre cualquier plataforma, siempre y cuando dichos servidores implementen la misma versión de *J2EE*.

La plataforma *J2EE* ofrece una serie de componentes y servicios que se pueden utilizar en el desarrollo de las aplicaciones y que resuelven gran parte de los problemas genéricos que se plantean a la hora de desarrollarlas. A continuación se mencionan algunos de los más importantes con un ejemplo de utilización:

- *Servlet*. Son objetos Java que extienden la funcionalidad de un servidor web. Mediante los Servlets se tratan las peticiones *HTTP* realizadas desde los navegadores.
- *JSP* (*Java Server Pages*). Es una tecnología que permite introducir código java dentro del código *HTML*. Mediante las *JSP*s se consigue independizar la capa de presentación del resto.
- *EJB*. Es una tecnología para desarrollar componentes en la parte del servidor. Son muy útiles debido a que pueden proporcionar transaccionalidad, portabilidad, seguridad, etc. a los desarrollos.
- *JMS (Java Message Service)* [52]. Tecnología que ofrece, mediante su *API*, servicios de mensajería típicos (Punto a punto y publicador-subscriptor) para comunicar aplicaciones, componentes, etc. Mediante el uso *JMS* se consiguen comunicaciones asíncronas.
- *JNDI* (*Java Naming and Directory Interface*). Servicio de nombrado de recursos y objetos. Mediante *JNDI* se pueden localizar recursos y objetos desplegados en el servidor local, y también en remotos.
- *JDBC* (*Java Data Base Connectivity*). Proporciona una *API* estándar para la comunicación con base de datos relacionales. Cada gestor de base de datos (oracle, *MYSQL*, *DB2*, etc.) ofrece siempre sus drivers a modo de librería que cumple con el *API* de *JDBC*.
- JavaMail. Este *API* proporciona el interfaz necesario para la gestión del correo electrónico.

La arquitectura *J2EE* está basada en el modelo componente-contenedor, donde los componentes son *JSP*'s, *Servlets* y *EJB*'s y el contenedor es el servidor donde se despliegan los componentes. En la Figura 3.1 se representa de forma gráfica la arquitectura *J2EE*.

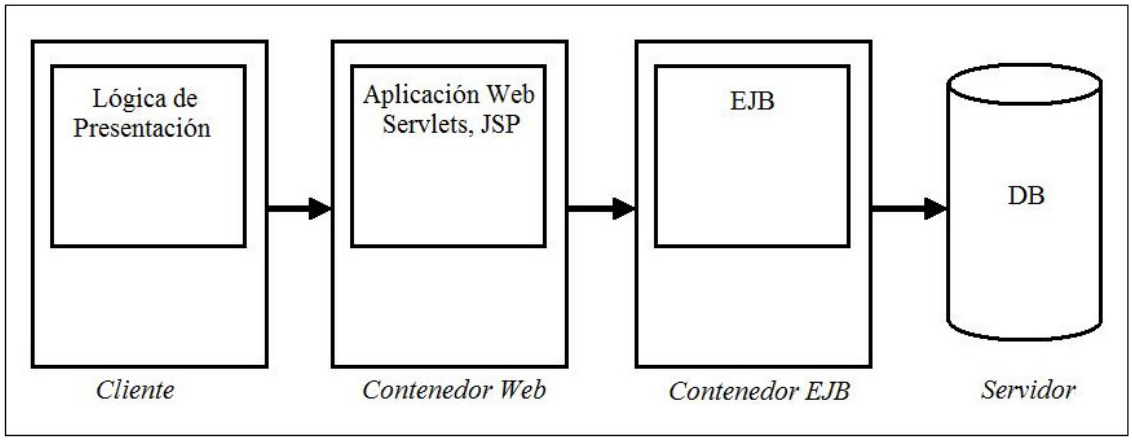

Figura 3.1: Arquitectura de *J2EE*.

Una vez creada la aplicación deseada, se necesita un servidor de aplicaciones para conseguir desplegar los componentes de dicha aplicación. En el mercado existen múltiples servidores de aplicaciones, los hay propietarios y de código libre y se clasifican por su capacidad de desplegar contenedores Web y contenedores *EJB*.

En el caso del presente proyecto, el servidor de aplicaciones escogido fue el denominado *Jboss*, que como ya se ha descrito antes, es un servidor de aplicaciones capaz de desplegar contenedores *EJB*.

- JSF.

JSF es un marco de trabajo para la creación de aplicaciones *J2EE* basadas en el modelo de la implementación de las páginas web que permiten la invocación de acciones determinadas mediante las peticiones realizadas por el usuario mediante la interacción con la vista presentada.

Dicho marco utiliza páginas *JSP* para generar las vistas y asocia a cada vista con formularios, un conjunto de objetos tipo *Bean*, lo que facilita la recogida, la manipulación y la visualización de los datos mostrados en los formularios correspondientes. *JSF* se integra dentro de la página *JSP* y se encarga de la recogida y la generación de los datos de los componentes de dicha página. Además permite introducir javascript en la página con el fin de acelerar la respuesta de la interfaz del navegador del usuario.

JSF permite crear de forma rápida aplicaciones de negocio dinámicas en las que toda la lógica de negocio se implementa en java o es invocado desde java. La principal función del marco *JSF* es asociar a las vistas, clases java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario. Se consigue resolver de manera muy sencilla y automática muchas tareas, algunas de las cuales se enumeran a continuación:

- Recoger los datos introducidos por el usuario en los campos del formulario.
- Mostrar datos al usuario en cajas de texto y tablas.
- Controlar el estado de los controles del formulario según el estado de la aplicación.
- Realizar validaciones y conversiones de los datos introducidos por el usuario.
- Controlar los eventos que ocurren debidos a la acción del usuario, como pueden ser las pulsaciones de teclas o botones y movimientos del ratón.

Las aplicaciones *JSF* están formadas por los siguientes elementos principales:

- Páginas *JSP* que incluyen los formularios *JSF*. Estas páginas son las encargadas de generar las vistas de la aplicación.
- Beans java que se conectan con los formularios *JSF*.

- Clases java para la lógica de negocio y utilidades.
- Ficheros de configuración.

- JMS.

En las comunicaciones entre cliente y servidor, los datos que se intercambian entre las dos partes, necesitan de una comunicación síncrona, es decir, que las dos partes estén presentes en el momento de la comunicación. Los sistemas de mensajes aportan una serie de mejoras a la comunicación entre aplicaciones que no tienen por qué residir en la misma máquina.

JMS es la solución de Sun para los sistemas de paso de mensajes. *JMS* se sitúa como la capa intermedia de la comunicación entre dos aplicaciones. La ventaja del sistema de mensajes reside en que la aplicación destino no tiene que estar conectada en el momento en el que la aplicación origen realiza el envío. La aplicación origen envía la información, el sistema de mensajes la almacena y se la transmite a la aplicación destino cuando ésta se conecte al servicio.

Existen dos modelos de mensajería:

- **Queue:** Se trata del modelo punto a punto. En este caso existe uno o varios publicadores y un único consumidor. Los generadores van dejando sus mensajes en la cola y el consumidor los va recogiendo y posteriormente descartando. Si el consumidor no se encuentra disponible, en esta modalidad de paso de mensajes, la información se va almacenando en la cola, de forma que el consumidor puede retomar su procesamiento una vez vuelva a estar disponible.

- **Topic:** En esta ocasión se trata de un modelo publicador-suscriptor. Es decir, en este caso existe un único publicador y uno o varios suscriptores que van recibiendo los mensajes que transmite el primero. Otra diferencia reside en el hecho de que el mensaje se pierde en caso de recibirse mientras el consumidor no se encuentra conectado.

En este proyecto se empleó el modelo de mensajes *Topic* debido a que los mensajes que contienen la información sobre los eventos de *Asterisk* ocurridos deben estar disponibles para varios posibles suscriptores.

- PostgreSQL.

Una de las partes importantes del proyecto recae en el manejo de las bases de datos ya que como ya se ha comentado, se utiliza *Asterisk realtime* el cual obtiene la información necesaria sobre la configuración de los registros de la base de datos.

En este proyecto se ha empleado el motor de base de datos denominado *PostgreSQL* que es un gestor de base de datos de código libre.

Una de las ventajas de este producto con respecto a otros reside en que se distribuye bajo licencia *BSD*, que a diferencia de la *GPL*, permite usarlo para fines comerciales.

El presente motor de base de datos se empleó debido a sus siguientes características:

1. Durabilidad. Una vez que una operación es llevada a cabo, será persistente y no se verá afectada aunque ocurra un fallo de sistema.
2. Consistencia. Se garantiza que sólo se ejecutan las operaciones que cumplen con las reglas y las directrices de la base de datos.
3. Atomicidad. Se asegura que ante un fallo de sistema ninguna operación puede quedar a medias de su ejecución.

4. Aislamiento. Se garantiza que dos transacciones distintas sobre la misma información no generarán ningún error ya que se asegura que una operación no puede verse afectada por otras.

Además, proporciona soporte de todas las características de una base de datos profesional como son los disparadores, las secuencias, las relaciones, las reglas, los tipos de datos definidos por usuarios, las vistas, etc. Es importante resaltar también que el máximo de bases de datos, a la vez que el máximo de registros por tabla, es ilimitado.

- Herramientas para la documentación.

Para la realización de la documentación de este proyecto se ha empleado Microsoft Office Word 2007 en el sistema operativo Windows.

Los diagramas de flujo han sido creados mediante el uso de Microsoft Office Visio 2007.

Para llevar a cabo la edición de imágenes empleadas en la memoria, se utilizó el programa Paint.

3.2.2. Equipamiento hardware.

A lo largo del desarrollo de la aplicación de Terminal de Operador ha sido necesario el empleo de distintos dispositivos hardware, los cuales se enumeran a continuación:

- Ordenador portátil modelo Aspire 5051 AWXMi.
  Equipo empleado para la implementación del código.

- Ordenador Pentium Core Duo, 2.4GHz.
  Equipo que alberga la centralita *PBX* basada en *Asterisk* modificado.

- Tarjetas.
  Para la configuración de la centralita se emplearon tanto tarjetas digitales como analógicas.
  - Xorcom Astribank 2 *BRI* (XR0013)
  - Xorcom Astribank 8 *FXO* (XR0019)
  - Xorcom Astribank 8 *FXS* (XR0001)

- Teléfonos *IP*.
  Para la realización del proyecto se emplearon varios modelos de teléfonos *IP*.
  - Thomson ST2030
  - Linksys IP Phone SPA942
  - Grandstream BT201
  - DI-Voz 50

- Teléfonos analógicos.
  Se emplearon diversos teléfonos analógicos a lo largo del desarrollo del proyecto pero a la hora de la redacción de la memoria no se disponía de la información sobre los modelos y fabricantes en concreto.

3.3. Diseño.

A la hora de enfrentarse a la realización de un proyecto de cierta envergadura, es imprescindible dedicar tiempo y todo tipo de recursos a la realización del diseño exhaustivo del mismo. A lo largo de la elaboración del proyecto, dicho esfuerzo se traducirá en un ahorro de todos los recursos empleados, siempre y cuando el diseño se realice de forma eficiente. Es por

tanto ineludible reflejar la importancia de esta etapa del desarrollo del proyecto efectuando una detallada descripción del proceso seguido.

El proceso de diseño está formado por numerosas etapas pero se puede realizar una distinción de dos bloques importantes. El primero de ellos resulta ser la especificación de las funcionalidades de la aplicación, es decir, establecer las capacidades de las que debe disfrutar el producto. Y el segundo bloque sería el diseño propiamente dicho, que sería decidir cuál es la mejor forma de implementar las funcionalidades deseadas. A lo largo de los apartados siguientes, se describe todo el proceso de diseño seguido durante el desarrollo del proyecto.

3.3.1. Funcionalidad requerida.

El propósito del presente proyecto reside en construir una aplicación que permita llevar a cabo la monitorización de una centralita basada en *Asterisk* y que esté preparada para permitir al usuario encargado de realizar dicha monitorización, interactuar con la centralita a través del interfaz desarrollado.

Una de las etapas iniciales del desarrollo de un proyecto es la construcción de la especificación de las funcionalidades de las que debe disponer. Para ello se realizó un análisis exhaustivo de las expectativas existentes y de los recursos disponibles y establecieron las especificaciones que debía respetar el producto que aparecen descritas a continuación.

- Tras la realización de la autenticación del usuario, permitir que éste acceda a sus datos personales y que recupere toda la información relevante que haya podido almacenar en conexiones anteriores. Una vez recuperada dicha información, proporcionarle el interfaz personalizado preparado para el desarrollo de su actividad.
- Construir una interfaz para que el usuario pueda realizar la configuración de la aplicación mencionada en el apartado anterior. Para ello se plantearon varios puntos que se consideraron relevantes.

    - Las extensiones del sistema que se deseen monitorizar deben ser seleccionables y el usuario encargado de la monitorización debe poder disponer de la posibilidad de distribuirlas de forma más cómoda para la realización de su tarea.
    - Debe ser posible la realización de la configuración de las pantallas para que el usuario pueda seleccionar las que crea que son más útiles para su trabajo.
    - La identificación de las extensiones del sistema debe poder realizarse tanto por *login* del usuario como por su nombre y apellidos para una identificación más eficiente.
    - El usuario debe disponer de la posibilidad de modificar la configuración de su terminal en cualquier momento.
    - El usuario tiene que tener un fácil acceso a su agenda telefónica, para modificarla en cualquier momento y emplearla en su labor.

- Permitir la monitorización de todos los componentes clave de la centralita, con este fin se planteó que sería necesario visualizar el estado de las extensiones consideradas relevantes y de las líneas analógicas y digitales del sistema. Para ello se decidió disponer de un cuadrante que mostrara en todo momento el estado de las extensiones asociadas a los usuarios y un segundo cuadrante con el estado de las líneas telefónicas del sistema. El cuadrante de las extensiones debía cumplir ciertas características:

    - Reflejar en todo momento el estado de las extensiones monitorizadas.
    - Las extensiones pueden ser tanto analógicas como digitales.
    - En caso de que un usuario se encontrara ocupado con una conversación, mostrar el número de teléfono o el identificador del interlocutor.
    - Suplir las posibles deficiencias del terminal telefónico del usuario.

- Permitir el establecimiento de una llamada con la extensión escogida desde el terminal.
- Permitir la transferencia de una llamada entrante o saliente del sistema hacia la extensión escogida mediante la interacción con el interfaz.

Para una completa monitorización de las líneas del sistema, se consideraron importantes los siguientes puntos:

- Reflejar si las líneas se encuentran operativas, es decir, si existe una conexión física que permite su correcto funcionamiento.
- Mostrar qué líneas se encuentran ocupadas.
- Permitir el establecimiento de una llamada encaminándola hacia una línea de salida en concreto mediante la interacción con la agenda del usuario.

- Una vez conseguida una fácil administración de la agenda telefónica del usuario, se planteó la necesidad de implementar la posibilidad de interacción con la susodicha desde el interfaz, se consideró que sería interesante permitir al usuario:

  - Realizar el establecimiento de una llamada con una entrada de la agenda desde el interfaz.
  - Permitir realizar una transferencia de una llamada entrante o saliente del sistema directamente hacia un número de la agenda.
  - Proporcionar la posibilidad de realizar una búsqueda del contacto deseado entre los disponibles en la agenda.
  - Conseguir establecer una llamada desde el interfaz con un número de teléfono cualquiera, aunque no estuviera registrado en la agenda.

- Uno de los puntos más importantes residía en conseguir una satisfactoria monitorización de las llamadas entrantes y salientes del sistema, para ello se planteó reservar otro cuadrante que debía cumplir con:

  - Reflejar la generación de una nueva llamada del sistema ofreciendo datos sobre el origen y destino de la misma. Se consideró interesante que en caso de que el interlocutor no perteneciente al sistema estuviera registrado en la agenda del usuario, que fuera identificado por su nombre. En caso contrario simplemente se mostraría el número de teléfono involucrado.
  - Ofrecer un contador de la duración de la llamada desde el momento de la conexión física de los dos interlocutores.
  - Permitir la transferencia de las llamadas, tanto entrantes como salientes mediante la interacción con el interfaz. Y a su vez, que dicha transferencia sea posible tanto hacia una extensión del sistema como hacia un contacto reflejado en la agenda.
  - Permitir la priorización de una llamada entrante, es decir, escoger la llamada que se desea atender en todo momento mediante la interacción con el interfaz.
  - Recuperar cualquier llamada puesta en segundo plano cuando se desee.
  - Disponer de la información sobre las llamadas retenidas.

Tras el planteamiento realizado sobre las funcionalidades, que se pretendía que tuviera la aplicación, se procedió al diseño del proyecto para conseguir cumplir todos los puntos propuestos.

3.3.2. Decisiones de diseño.

Este apartado de la memoria explica y justifica las decisiones de diseño más importantes realizadas a lo largo de la ejecución del proyecto.

El primer gran problema al que hubo que enfrentarse al comenzar con el desarrollo de la aplicación fue decidir la estructura que debía respetar para conseguir la monitorización de la centralita en tiempo real y que ésta se realizara de manera lo más eficiente posible. Hubo dos opciones más relevantes que se barajaron al principio del desarrollo. Una de las opciones consistía en realizar la recepción de los eventos que describen el comportamiento de la centralita y almacenar la información relevante en una base de datos. A continuación, la parte web se encargaría de realizar peticiones periódicas a la base de datos para presentarle al usuario el estado actualizado del sistema.

La segunda opción consistía en construir la aplicación de forma que la página web solicitara información actualizada y permaneciera a la espera hasta que dicha información estuviera disponible. Finalmente se optó por esta segunda posibilidad ya que aunque presentaba algunos problemas de sincronización, consumía menos recursos y por tanto resultaba ser mucho más eficiente.

A la hora de comenzar el proceso de diseño del producto de Terminal de Operador, hubo que enfrentarse al hecho de que ciertos aspectos del producto ya se encontraban definidos ya que éste formaba parte de un proyecto mayor. Es decir, por ejemplo, se tuvo que emplear el mismo diseño de página de acceso de usuario, aunque el diseño de la interfaz de trabajo sí que se elaboró de cero. Sin embargo también había que cumplir con ciertas restricciones ya que la página debía respetar los colores corporativos.

Una vez identificadas las limitaciones de libertad con respecto a la visualización del interfaz se procedió a la definición de la información relevante a monitorizar. Siguiendo el apartado anterior, en el que se describen las funcionalidades que se requerían del proyecto, se puede observar la lógica del diseño de la aplicación.

Para cumplir con las exigencias de funcionalidad, se decidió dividir la información presentada en cuatro grupos que se enumeran a continuación.

- Las extensiones del sistema.
- Las llamadas en curso.
- La agenda del usuario operador.
- Las líneas del sistema.

Con respecto a las llamadas del sistema, se consideró que las conversaciones entre los usuarios del sistema no tenían la misma relevancia que las llamadas entrantes o salientes. Para remarcar dicha diferencia, para representar una llamada interna del sistema, sólo se emplea la subpantalla de las extensiones. Es decir, la subpantalla de las llamadas del sistema se encarga de la presentación de las llamadas entrantes y salientes del sistema, mientras que una llamada interna se presenta como cambio de estado de las extensiones involucradas en la subpantalla de las extensiones.

En una etapa inicial del diseño se barajó la posibilidad de ofrecer al usuario un registro de llamadas recientes porque se consideró que podía ser de interés disponer de dicha información y ofrecer la opción de recuperar una llamada reciente. Sin embargo, tras profundizar en el diseño de la herramienta, se llegó a la conclusión de que disponer de la agenda de usuario y poder interactuar con ella resultaba mucho más útil y es ésta quien forma parte del diseño final.

Otra decisión de diseño importante incumbe al número de las pantallas presentadas al usuario. Desde el principio se decidió que debían ser cuatro para conseguir un producto con un diseño simétrico, ordenado e intuitivo para el usuario. No obstante, tras una primera etapa del desarrollo se descubrió imprescindible permitir que el aspecto presentado por la herramienta fuera configurable. Se decidió que podía ser interesante permitir al usuario eliminar algunas de las vistas disponibles para permitir dar prioridad a las subpantallas que éste considere más útiles para su trabajo. Una vez decidido proporcionar la posibilidad de configuración, se procedió al estudio de las posibilidades que serían interesantes de ofrecer. En esta etapa se decidió que la

distribución de las extensiones monitorizadas también debía ser configurable para facilitar al máximo la labor del usuario encargado de la tarea.

Con respecto a la tecnología escogida para llevar cabo el desarrollo, en principio también debía suponer una decisión de diseño muy importante sin embargo, como ya se ha comentado, debido al hecho de que el presente producto forma parte de un proyecto mayor, había varios aspectos que venían impuestos por esta razón. Se tuvo que emplear la tecnología de *J2EE* explicada en el apartado de herramientas pero sin suponer ningún detrimento ya que ésta ofrece todas las facilidades posibles para la elaboración de un producto como el descrito en la presente memoria.

Durante la etapa del inicio del diseño se pensó en las posibles ventajas que podía ofrecer la implementación de la herramienta como un ejecutable en lugar de una aplicación web. Tras diversas consideraciones, se decidió que la herramienta debía formar parte del proyecto total y por lo tanto debía ser accesible a través de la web. Sin embargo, se planteó como una mejora a largo plazo, su implementación como ejecutable para abarcar una franja de mercado mayor satisfaciendo las necesidades de más clientes.

3.3.3. Arquitectura de la aplicación.

El propósito de este apartado reside en presentar la arquitectura de la aplicación describiendo las diferentes unidades lógicas por las que está formada y la relación que existe entre ellas. En la Figura 3.2 se representa dicha arquitectura de forma gráfica, mientras que el vínculo entre los distintos componentes se describe a lo largo de los párrafos siguientes ante la imposibilidad de representar gráficamente dichas relaciones de forma clara.

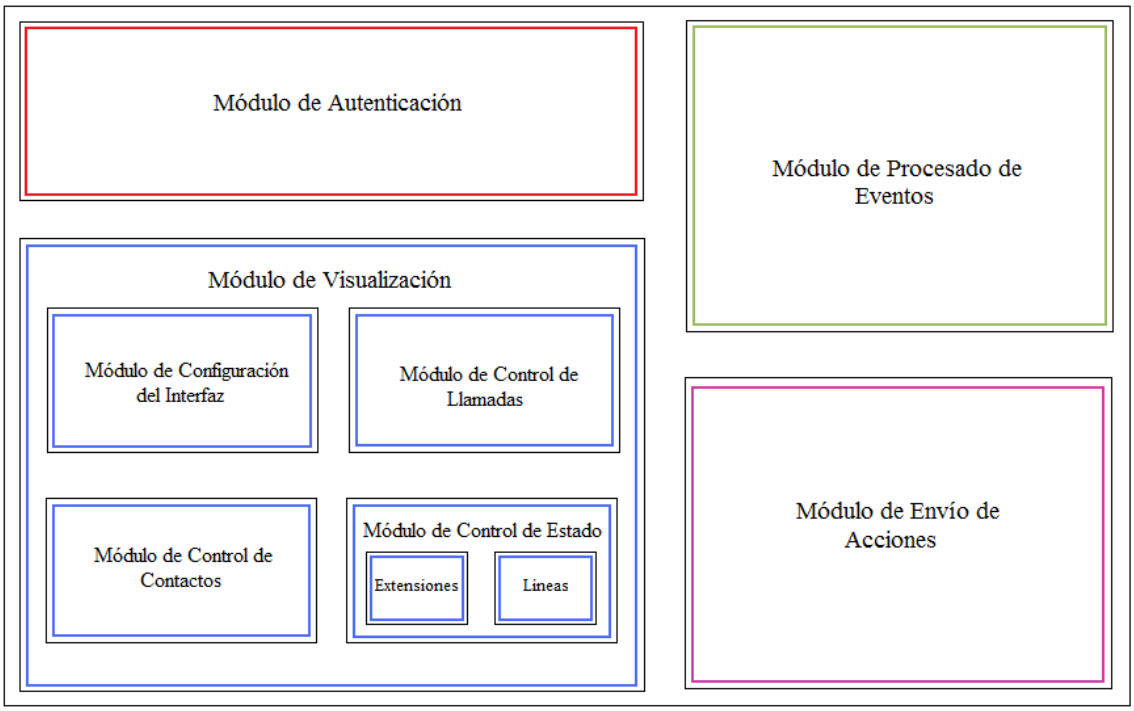

Figura 3.2: Arquitectura de la aplicación "Terminal de Operador".

3.3.3.1. Módulo de autenticación.

La presente parte de la aplicación se encarga de la autenticación del usuario que está intentando acceder al interfaz de operador. Para empezar, se describirán brevemente los tipos de usuario que permite el sistema.

La centralita telefónica se diseñó de forma que pudiera tener distintos tipos de usuario que disfrutaran de diferentes privilegios o permisos dependiendo de su responsabilidad dentro de la administración de dicha centralita. Lógicamente existe un tipo de usuario administrador que posee permisos para acceder a un menú especial para los supervisores. Dicho menú dispone de funcionalidades que permiten realizar una configuración completa de la centralita.

También existe un tipo de usuario que simplemente representa a todos los miembros de la organización que vaya a hacer uso de la centralita. Dichos usuarios no disponen de autorización para realizar ningún cambio en la configuración de la centralita ni pueden alterar su funcionamiento de ningún modo. Tan sólo les está permitido crear y administrar su propia agenda, modificar su contraseña de acceso a la aplicación y visualizar el directorio en el que se almacenan los datos de todos los miembros de la organización.

Por último, existe un tipo de usuario operador, el cual disfruta de todos los servicios descritos en el apartado anterior pero además dispone de permisos para realizar una monitorización del sistema. Es decir, este tipo de usuario puede construir su propio subconjunto de usuarios cuyo comportamiento desea monitorizar y controlar.

La función básica del módulo de autenticación reside en, a partir del identificador de usuario y su contraseña de acceso proporcionados por éste, averiguar de qué tipo de usuario se trata y mediante la interacción con un módulo externo de base de datos, obtener la información necesaria sobre su perfil.

Si el proceso del acceso a la base de datos tiene éxito y realiza correctamente la identificación del usuario que intenta el acceso a la aplicación, el módulo de autenticación permite el progreso del resto de los módulos que forman parte del módulo de visualización para que éstos de encarguen de presentarle al usuario el interfaz correspondiente a su perfil. En caso contrario, si la base de datos no dispone de datos correspondientes al usuario que trata de realizar el acceso, o se ha producido algún error en la introducción de las credenciales, el módulo de autenticación se encarga de mostrarle al usuario una ventana informando de que ha habido un error de autenticación denegando el acceso del usuario al sistema.

3.3.3.2. Módulo de visualización.

A continuación se describirá el módulo de visualización, el cual comprende toda la parte relativa a la interfaz web del proyecto. Como se puede observar en la ilustración presentada en la Figura 3.2, la herramienta está formada por otros dos módulos del mismo nivel, aparte del módulo de autenticación. La interacción básica existente entre estos tres componentes reside en que el módulo de visualización se encarga de realizar la presentación al usuario de la información obtenida por el módulo de procesado de eventos, y a su vez, recoge las demandas del usuario para transmitírselas al módulo de envío de acciones. Dicho módulo de visualización está formado por distintos componentes que se pasarán a detallar a continuación.

3.3.3.2.1. Módulo de configuración del interfaz.

El siguiente módulo es el encargado de prestar la funcionalidad de la posibilidad de realizar una configuración del interfaz de operador que permite personalizar la monitorización para que el usuario pueda realizar su labor de la manera más cómoda y eficiente para él, dentro de las posibilidades existentes.

Para llevar a cabo dicha configuración, se diseñó y se construyó una nueva interfaz web que se encuentra disponible desde la vista de usuario tipo operador.

El presente módulo se encarga de comunicarse con el módulo externo encargado de la gestión de la base de datos para comprobar si ya existe alguna configuración previa para el usuario operador que está accediendo a la página de configuración. Para ello, el módulo de configuración está encargado de mantener almacenada la identificación del usuario desde el momento de su acceso a la aplicación para poder hacer uso de dicha información de cualquier momento.

Una vez realizada la comunicación con la base de datos, el módulo de configuración actúa en consecuencia dependiendo de la información obtenida. En el caso de que ya existiera una configuración anterior, se analiza el informe elaborado a partir de las entradas de la base de datos y se presenta al usuario el resumen del mismo. La otra opción es que sea la primera vez que el usuario accede a esta interfaz, y por lo tanto, en la base de datos todavía no se encuentra reflejada ninguna configuración anterior. En este caso, el módulo de configuración se encarga de presentarle al usuario la vista por defecto para que éste pueda elegir cómo desea que le sea presentada la información de la monitorización de la centralita.

En el momento del acceso a la base de datos que se ha comentado en el párrafo anterior, aparte de intentar conseguir información sobre una posible configuración previa, se averiguan también los detalles relevantes de todas las extensiones configuradas en el sistema. Los cuales se presentan a continuación al usuario para que éste pueda escoger a cuáles de las disponibles desea monitorizar.

A continuación, el usuario ya se encuentra en disposición de realizar una nueva configuración o modificar la existente. El módulo de configuración le permite al usuario escoger las extensiones del sistema que éste desea monitorizar y además le proporciona la posibilidad de organizarlas como desee en la parrilla de presentación. Además, el usuario puede escoger entre diferentes configuraciones de pantallas disponibles y decidir cómo quiere que sean identificados los usuarios monitorizados. Toda la información sobre la configuración realizada es recibida por el módulo de configuración y enviada al módulo de la base de datos para su almacenamiento.

En cualquier momento, el usuario dispone de la posibilidad de acceder al interfaz de configuración para visualizar la configuración existente y modificarla de la manera que él desee.

3.3.3.2.2. Módulo de control de contactos.

A continuación se describe el módulo que se encarga del control de la agenda del usuario operador.

Para llevar a cabo la monitorización del sistema, el usuario operador tiene a su disposición la posibilidad de visualizar la agenda telefónica en la que él mismo introduce, modifica o suprime entradas. El presente módulo se encarga de gestionar todas las operaciones que puedan realizarse con o sobre las entradas registradas en dicha agenda.

En un estado inicial, el módulo de control de contactos se encarga de presentar al usuario operador un listado con los contactos que dicho operario haya insertado en su agenda a través de otra página web accesible desde su vista de usuario.

Para en el caso en que la extensión de la agenda sea considerable, el módulo de control de contactos permite realizar una búsqueda de la entrada deseada o de las entradas relacionadas. Es decir, el usuario operador, introduciendo una palabra o simplemente unas letras en el espacio reservado a tal efecto, recibe por pantalla una colección de las entradas de su agenda que cumplan con el patrón introducido.

Cuando el contacto deseado ya ha sido localizado, ya sea mediante la ayuda de la búsqueda o a simple vista, el usuario operador dispone de la posibilidad de iniciar una llamada con dicho contacto. Para ello no tiene más que realizar un doble click sobre la entrada de la agenda deseada y el módulo de control de los contactos se encarga de realizar la petición de establecimiento de una nueva llamada desde el terminal telefónico del usuario operador hacia el número de teléfono que corresponde al contacto seleccionado. El módulo responsable de la operación se encarga también de diferenciar el contacto involucrado en la operación para que al usuario le resulte más fácil y para que no tenga dudas sobre si ha pulsado la entrada correcta. Para ello, la entrada de la agenda con la que se está intentando establecer la llamada aparecerá parpadeando y cambiará de color para conseguir el efecto descrito.

Si el usuario operador desea comunicarse con un número de teléfono que no se encuentra almacenado en su agenda, puede hacerlo perfectamente también desde el interfaz simplemente introduciendo el número en el espacio reservado a tal efecto y pulsando el botón de marcado. El proceso que se sigue en este caso es exactamente el mismo sólo que la fuente del número de teléfono del interlocutor es diferente.

3.3.3.2.3. Módulo de control de llamadas.

A continuación se va a describir la funcionalidad del módulo que se encarga del control de las llamadas presentes en el sistema. Como se comentó en la parte de la descripción del diseño en la sección 3.3.2, se consideró que las llamadas internas del sistema no fueran relevantes a la hora de la monitorización de las llamadas del sistema. Es decir, en el caso de las llamadas internas, la recogida de la información pertinente recaía sobre el módulo encargado del estado de las extensiones. La aplicación se diseñó de forma que el módulo de control de llamadas se encargaba de gestionar sólo las llamadas entrantes y salientes del sistema.

En el momento inicial de acceso a la interfaz de monitorización, el módulo de control de llamadas necesita establecer una comunicación con el *Asterisk* para realizar una consulta sobre si existe alguna llamada establecida y para obtener la información pertinente sobre el estado de los usuarios involucrados. Para ello es precisa la interacción con el módulo de envío de acciones para realizarle la petición del informe sobre el estado de la centralita.

El módulo encargado del envío, realiza el análisis de la respuesta recibida por parte de *Asterisk* y en caso de que exista alguna llamada en curso en ese momento inicial, realiza la construcción de la estructura pertinente y le comunica la situación actual al módulo de control de llamadas. El cual efectúa la representación de la llamada en el interfaz de la aplicación.

Una vez que ya se ha concluido la parte de la inicialización, el módulo que nos ocupa se encarga de mantener actualizada la información sobre las llamadas presentes en el sistema. El usuario operador dispone de la posibilidad de interactuar con el sistema realizando diversas acciones sobre las llamadas entrantes y salientes. Dichas peticiones del usuario son atendidas también por el módulo de control de llamadas. Para llevar a cabo, por ejemplo, una transferencia de llamada, el módulo de nuevo se comunica con el encargado del envío de acciones para expedir una petición a *Asterisk* de modificar uno de los interlocutores ocupados en la conversación escogida. El descrito cambio se traduce en la generación de una colección de nuevos eventos que son analizados por el módulo de procesado de eventos descrito más adelante, en la sección 3.3.3.3. El módulo de control de llamadas recibe la información actualizada sobre el desarrollo de la llamada afectada y se encarga de presentársela al usuario.

El proceso de actuación sigue la misma estructura para cualquier acción del usuario que implique las llamadas entrantes o salientes del sistema.

3.3.3.2.4. Módulo de control del estado.

El siguiente componente de la aplicación es el encargado de la monitorización del estado tanto de las extensiones del sistema, como de las líneas analógicas y digitales disponibles.

La función básica de este módulo es permanecer a la escucha de la información actualizada sobre el estado del sistema que le proporciona el módulo de procesado de eventos ya que cuando se produce un cambio en el estado de un componente, *Asterisk* se encarga de generar un evento avisando de lo ocurrido.

Este módulo está formado por dos partes diferenciadas, una que trata el estado de las extensiones y la segunda que se encarga del estado de las líneas analógicas y digitales presentes en el sistema.

3.3.3.2.4.1. Estado de las extensiones.

Para explicar esta parte de la funcionalidad se comenzará con una breve introducción explicando la forma que tiene *Asterisk* de distinguir el estado de un usuario del sistema.

En *Asterisk*, un usuario del sistema se denomina "*peer*", cuando se trata como receptor de las llamadas tramitadas por *Asterisk* y básicamente corresponde a una extensión dada de alta y que se comunica mediante el protocolo *SIP*. Se definen 8 posibles estados para esos usuarios y son los siguientes:

- Estado 0, que corresponde a un estado desconocido del *peer*.
- Estado 1, el *peer* no está siendo usado por *Asterisk*.
- Estado 2, el *peer* está siendo usado por *Asterisk*.
- Estado 3, el *peer* se encuentra ocupado.
- Estado 4, corresponde a un estado no válido dentro del sistema.
- Estado 5, el *peer* no se encuentra disponible.
- Estado 6, denota que el terminal registrado en la extensión correspondiente al *peer* está sonando.
- Estado 7, estado que describe la situación en la que el *peer* se encuentra ocupado con una llamada y le está entrando una nueva.
- Estado 8, cuando se opta por la opción de retener la llamada.

Basándose en la información presentada y analizando las necesidades de los usuarios de la aplicación, se definieron 5 estados diferentes encargados de describir las condiciones en las que se encuentra un usuario del sistema y se describen a continuación:

- Libre.

Este primer estado corresponde a la situación en la que el peer en cuestión no se encuentra involucrado en ninguna llamada y ninguna otra actividad como puede ser formar parte de una sala de conferencia o haber retenido alguna llamada en curso.

- Ocupado.

Un usuario se encuentra en la situación definida para este estado en el momento en el que se realiza la unión del canal reservado al peer y del canal asignado a su interlocutor. Es decir, al usuario se le asignará este estado tanto si se encuentra interviniendo en una llamada con el exterior, como si lo está en una interna o si se encuentra participando en una sala de conferencias.

- Sonando.

Se definió que el estado de un peer se designará como sonando en caso de que al usuario le esté entrando una llamada cuando justo en el instante anterior se encontraba libre o todas sus llamadas iniciadas se encuentran retenidas. Hay que señalar el hecho de que el estado de Ocupado prevalece sobre el de Sonando debido a una decisión de diseño, es decir, si un usuario se encuentra ocupado en una conversación y le entra una nueva llamada, su estado seguirá siendo Ocupado.

- No disponible.

El presente estado describe la situación del usuario en caso de que su terminal no se encuentre correctamente registrado en el sistema. Dicha situación se puede dar por varias causas como por ejemplo problemas de red o un fallo en la alimentación del terminal en caso de que no disponga de *PoE* (*Power over Ethernet*).

- Retenidas.

Por último se definió este estado para describir aquella situación en la que el usuario correspondiente al peer implicado tenga retenidas todas las llamadas en las que participa y todavía se encuentran activas. Se consideró relevante definir un estado diferente para describir dicha situación ya que a la hora de realizar pruebas iniciales para familiarizarse con el comportamiento de la centralita, se observó que la forma de anunciar la existencia de una llamada retenida dependía de cada modelo de terminal que en algunos casos podría considerarse deficiente. Por esta razón se decidió añadir dicho estado para intentar independizar la aplicación de los modelos de terminal telefónico empleados y facilitar el trabajo del usuario de la aplicación.

El presente módulo recibe la información elaborada por el módulo encargado del análisis de los eventos enviados por *Asterisk* y se encarga de representar la extensión en el panel de la monitorización con la imagen que caracteriza el estado correspondiente.

Otra funcionalidad del presente módulo reside en la ejecución de los procesos necesarios para responder a las peticiones del usuario operador referidas a las extensiones del sistema. Por ejemplo, el usuario dispone de la posibilidad de establecer una llamada con una de las extensiones del sistema efectuando con el ratón un doble click sobre la imagen que representa al usuario escogido.

El módulo del estado de las extensiones se encarga de recoger la petición del usuario que está realizando la monitorización del sistema e interactúa con el módulo de envío de las acciones para efectuar la demanda del establecimiento de una llamada interna con el usuario escogido. Dicha petición provoca la generación de diversos eventos que describen el cambio de estado del sistema. El módulo de procesado de eventos elabora un informe sobre la situación actual y el módulo de estado de las extensiones se encarga de presentarle al usuario la información actualizada.

El proceso de actuación es similar ante cualquier otra petición del usuario operador sobre las extensiones del sistema.

3.3.3.2.4.2. Estado de las líneas.

Este módulo tan sólo tiene relevancia en el caso de que la conexión de las líneas a la centralita *Asterisk* se realice mediante tarjetas en lugar de un router, que son las dos opciones posibles. Para explicar la funcionalidad de este módulo se asumirá que se dispone de un sistema conectado al mundo exterior mediante tarjetas. En el caso de que la monitorización del estado de las líneas sea irrelevante, la aplicación ofrece la opción de eliminar la correspondiente subpantalla del interfaz de terminal de operador.

El presente módulo se encarga de mantener actualizado el estado de las líneas presentes en el sistema. En el momento inicial de acceso a la aplicación, este módulo establece una comunicación con *Asterisk*, mediante la intervención del módulo de envío de acciones, para consultar los datos disponibles de la configuración de las tarjetas conectadas al sistema. Gracias al análisis de dicha información se obtiene el número de líneas realmente habilitadas para la actividad de la centralita, que no tiene por qué coincidir con el número total de líneas de las tarjetas. Por tanto, sólo se monitorizarán las líneas realmente configuradas para su utilización. A continuación, a lo largo de la actividad normal de la centralita, el presente módulo se encargará de mantener actualizado el estado de las líneas, mostrando al usuario operador qué línea en concreto se está usando en cada momento y advirtiéndole de si ha habido algún fallo en la conexión física del cable correspondiente.

El usuario operador dispone de la posibilidad de encaminar una llamada saliente por una línea de salida en concreto, funcionalidad proporcionada por el módulo que se está describiendo. El usuario puede seleccionar un número de la agenda o introducir un número cualquiera no registrado en el sistema, en el espacio reservado a tal efecto, y arrastrarlo mediante el manejo del ratón hasta el icono que representa la línea por la cual desea que se encamine la llamada. En tal caso, el presente módulo se comunica con el módulo de envío de acciones para que éste mande a *Asterisk* la orden pertinente para conseguir el efecto deseado.

3.3.3.3. Módulo de procesado de eventos.

Una vez analizada la parte de la aplicación encargada de la presentación de la información al usuario, en el presente apartado se explicará detenidamente la comunicación con la centralita *Asterisk* para la realización del análisis de su estado. Como el nombre mismo del módulo indica, su función principal reside en el procesado de los eventos recibidos para la obtención de un informe fehaciente sobre el estado de la centralita.

Como ya se ha comentado anteriormente a lo largo de este documento, una de las partes principales de este proyecto es la comunicación con *Asterisk* para la recepción de los eventos que éste genera como resultado de la actividad de la centralita. Es precisamente el procesado de dichos eventos, lo que permite a la aplicación presentar el estado actualizado de los componentes del sistema en todo momento.

El presente módulo se encarga de la escucha permanente de la actividad de *Asterisk*, el cual refleja toda su actividad mediante la ayuda de la generación de eventos. Cada evento generado se traduce en una estructura determinada que recoge toda la información pertinente para la caracterización del mismo. Dicha estructura posee campos comunes a todos los tipos de eventos como son por ejemplo el tipo o la fecha de generación, y su vez dispone de campos que son característicos y diferentes para cada tipo. A continuación se puede observar una muestra de la actividad de la centralita reflejada por los eventos.

```
NewChannelEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00
2009,privilege='call,all',calleridnum='usuario',calleridname='Julia
Pruebas',callerid='usuario',uniqueid='1261139921.20',state='Down',timestamp=
'1.26113992108484E9',channel='SIP/usuario-09a0c4d8',systemHashcode=12358012]

NewStateEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00
2009,privilege='call,all',calleridnum='usuario',calleridname='Julia
Pruebas',callerid='usuario',uniqueid='1261139921.20',state='Ring',timestamp=
'1.261139921088206E9',channel='SIP/usuario-09a0c4d8', systemHashcode=
32383760]

NewExtenEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00 2009,
privilege='call,all', extension='0664331336',
appdata='/srv/itd/Grabaciones/0664331336-usuario-Fri Dec 18 13:38:41
2009.wav', uniqueid='1261139921.20', context='12B01',
timestamp='1.261139921157721E9', channel='SIP/usuario-
09a0c4d8',priority='1',application='MixMonitor',systemHashcode=3426887]

NewExtenEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00 2009,
privilege='call,all', extension='0664331336', appdata=
'SIP/VyDA_RDSI/664331336', uniqueid='1261139921.20', context='12B01',
timestamp='1.261139921235583E9',channel='SIP/usuario-09a0c4d8',priority='2',
application='Dial', systemHashcode=30385106]

NewChannelEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00 2009,
privilege='call,all', calleridnum='<unknown>', calleridname='<unknown>',
callerid='<unknown>',uniqueid='1261139921.21',state='Down',timestamp='1.2611
3992123598E9',channel='SIP/VyDA_RDSI-09a1a0b8',systemHashcode=20973025]

DialEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00 2009,
privilege='call,all',destination='SIP/VyDA_RDSI-09a1a0b8',
calleridname='Julia Pruebas', callerid='usuario',
srcuniqueid='1261139921.20', src='SIP/usuario-09a0c4d8',
timestamp='1.261139921241968E9',destuniqueid='1261139921.21',systemHashcode=
8845383]
```

```
NewCallerIdEvent[dateReceived=Fri Dec 18 13:38:47 GMT+01:00 2009,
privilege='call,all', calleridnum='0664331336', cidcallingpres='0',
calleridname='<Unknown>',callerid='0664331336',uniqueid='1261139921.21',cidc
allingprestxt='Presentation Allowed, Not Screened',
timestamp='1.261139921292011E9',channel='SIP/VyDA_RDSI-09a1a0b8',
systemHashcode=23107602]

NewStateEvent[dateReceived=Fri Dec 18 13:38:54 GMT+01:00 2009,
privilege='call,all', calleridnum='0664331336', calleridname='<unknown>',
callerid='0664331336',uniqueid='1261139921.21',state='Ringing',timestamp='1.
261139927861944E9',channel='SIP/VyDA_RDSI-09a1a0b8',systemHashcode=21806911]

HangupEvent[dateReceived=Fri Dec 18 13:38:59 GMT+01:00 2009,
privilege='call,all',causetxt='Normal Clearing', calleridname='null',
cause='16', calleridnum='null', callerid='null', uniqueid='1261139921.21',
state='null',timestamp='1.261139933604745E9',channel='SIP/VyDA_RDSI-
09a1a0b8',systemHashcode=32908873]

HangupEvent[dateReceived=Fri Dec 18 13:39:00 GMT+01:00 2009,
privilege='call,all', causetxt='Unknown', calleridname='null', cause='0',
calleridnum='null', callerid='null', uniqueid='1261139921.20', state='null',
timestamp='1.261139933686822E9',channel='SIP/usuario-09a0c4d8',
systemHashcode=988057]
```

Cada vez que tiene lugar la generación de un evento, una de las parte del módulo lo recoge y se encarga de extraer los campos relevantes para su posterior análisis. A continuación, mediante el paso de mensajes, le envía la información recogida a la parte del módulo que realmente se va a encargar de analizarla y de extraer las conclusiones pertinentes. Por último, tras interpretar el significado del evento recibido, se envía la información actualizada sobre el estado del sistema al módulo de visualización, para ser presentada al usuario. Como ya se ha explicado con anterioridad, el módulo de visualización está formado por diversos componentes y cada uno de ellos se encarga de la presentación de la situación de la parte que le corresponde.

3.3.3.4. Módulo de envío de acciones.

El otro componente básico de la parte de control, es el segmento activo de la comunicación con *Asterisk*, es decir, en esta ocasión, el módulo es capaz de enviar órdenes y peticiones a la *PBX* para conseguir alterar su actividad o sencillamente solicitarle información.

El módulo cuya funcionalidad se está describiendo, entra en acción básicamente en dos situaciones. La primera tiene lugar cuando el usuario operador desea iniciar la monitorización del sistema y se hace necesaria la consulta del estado de la actividad de la centralita en ese preciso instante ya que en esa situación, el análisis de los eventos se hace insuficiente.

En la situación descrita intervienen todos los módulos encargados de la visualización excepto el correspondiente a la presentación de la agenda del usuario operador. Es decir, en el momento inicial del proceso, los submódulos de visualización demandan al módulo de envío de acciones que éste realice las solicitudes de información de estado pertinentes en cada caso.

La segunda situación en la que se hace imprescindible la intervención del módulo, es cuando el usuario operador desea realizar alguna intervención que implique la necesidad de la comunicación con *Asterisk*. Es decir, cuando por ejemplo desea transferir una llamada a un número registrado en su agenda, se hace necesario enviarle la orden pertinente a *Asterisk* para conseguir el efecto deseado.

En caso de que se detecte un intento de interacción del usuario operador con el sistema, el módulo correspondiente realiza una solicitud al módulo de envío de acciones para que éste a su vez, transmita la orden pertinente a la centralita. *Asterisk*, al recibir la orden, lleva a cabo operaciones que permiten la materialización de la acción impuesta por el usuario, dicha situación se refleja mediante la generación de eventos característicos. El módulo de procesado

de eventos se hace cargo de su parte correspondiente de trabajo y tras sacar la información pertinente de los eventos recibidos, se encarga de enviársela al submódulo de visualización oportuno.

3.3.4. Especificación.

En este apartado se realiza una explicación minuciosa de cada módulo que forma parte de la arquitectura de la aplicación con el nivel de detalle que debiera permitir la reproducción del trabajo realizado por una persona ajena al proyecto.

3.3.4.1. Módulo de autenticación.

Como ya se ha explicado en el apartado anterior, para la ejecución del proyecto se ha empleado la tecnología *JSF* que permite asociar un *bean* a una página web, de forma que en esta ocasión, en cooperación con el denominado *LoginServerBean*, se hace posible el acceso a la base de datos para comprobar si las credenciales del usuario que intenta acceder al sistema, realmente se encuentran registrados y a la vez se consulta el tipo de usuario del que se trata para conocer qué vistas está autorizado a visualizar.

En la Figura 3.3 se puede observar el diagrama de flujo correspondiente al módulo de autenticación.

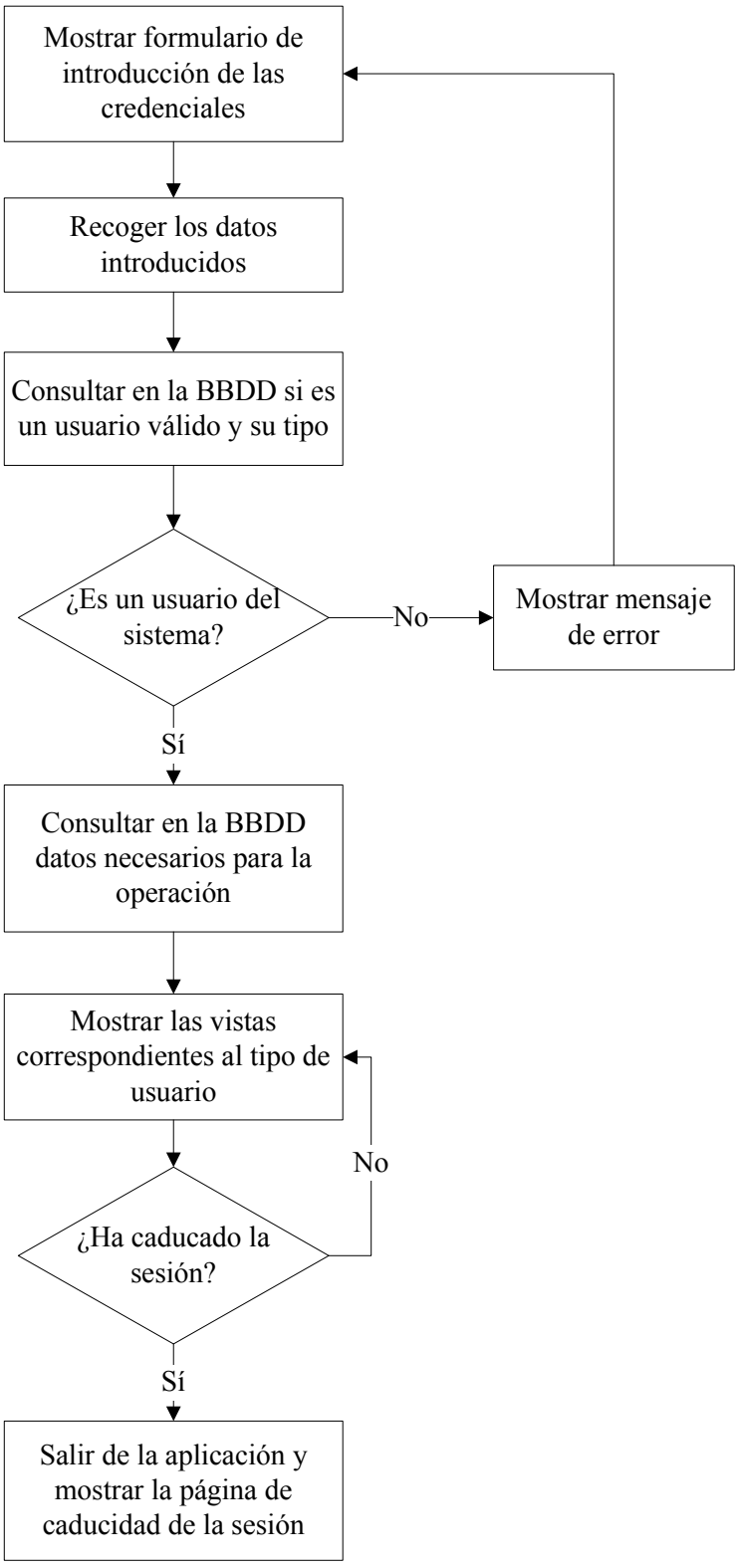

Figura 3.3: Diagrama de flujo del Módulo de Autenticación.

Tras la obtención del permiso para acceder al sistema en caso de que se trate de un usuario de tipo Operador, el siguiente paso es conseguir la información necesaria para permitir la interacción con el interfaz de monitorización del sistema.

En el momento en el que el usuario intenta visualizar la pestaña correspondiente al interfaz de operador, las tareas que se llevan a cabo se pueden observar en la Figura 3.4.

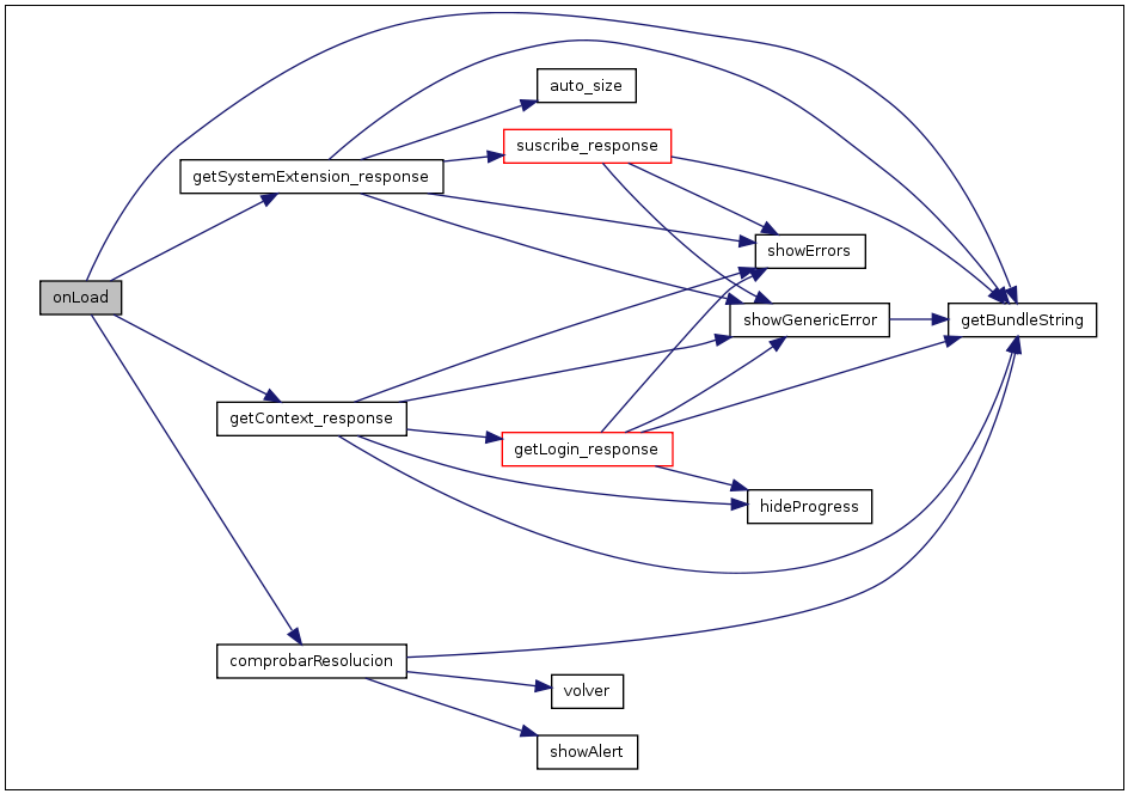

Figura 3.4: Grafo de llamada de la función de inicialización del Módulo de Autenticación.

Como se puede observar, el presente módulo se responsabiliza de inicializar valores imprescindibles para la actividad a través de la función de carga de la página del interfaz. En el capítulo de estado de arte, se explicaron brevemente las bases de funcionamiento de *Asterisk*, y según ésas, un usuario del sistema se caracteriza por un identificador denominado *login* y para permitir su actividad es necesario asociarle un contexto de trabajo. Para permitir la interacción con el sistema por parte del usuario operador, se hace necesaria la obtención de ciertos datos sobre su registro en el sistema, la responsabilidad de lo cual recae sobre el módulo de autenticación.

También debe comprobar si las características del equipo empleado para la monitorización, cumplen con las restricciones impuestas. Es decir, la aplicación está diseñada para un número de resoluciones acotado y si la resolución de la pantalla del equipo donde pretende desarrollar su actividad el usuario operador no se encuentra en el conjunto de las posibles, no se le permite el acceso a la pantalla de monitorización, mostrando un mensaje advirtiendo de la situación. Si la resolución resulta ser una de las permitidas, el aspecto de la pantalla se adapta a ésta de forma que la representación resulte ser óptima con respecto al aprovechamiento del espacio disponible.

Otra tarea importante del presente módulo consiste en inicializar la conexión de paso de mensajes necesaria para permitir la correcta actividad del módulo de procesado de eventos. A partir de este instante el módulo de procesado de eventos ya es capaz de recibir los eventos generados por la centralita.

En la Figura 3.4 se puede observar que también se hace uso de varias funciones auxiliares necesarias para permitir la funcionalidad descrita.

Para la implementación del acceso a la aplicación se ha empleado el denominado alcance de sesión, que permite al usuario la visualización de la página de inicio desde el

momento de su autenticación hasta que excede el tiempo de inactividad predeterminado para añadir seguridad al sistema.

Es importante resaltar que el mecanismo de seguridad descrito no se ha empleado en el caso de la página de monitorización del terminal. Dicha decisión se debe al hecho de que la monitorización puede ser muchas veces pasiva, el operador puede estar trabajando controlando el funcionamiento de la centralita pero no interactuar con la página porque no necesite alterar la actividad del sistema. Se consideró que si la página del terminal tuviera caducidad, podría suponer una molestia para el operador y por tanto no se implementó.

También hay que mencionar que la responsabilidad de liberar recursos en caso de desconexión recae sobre el módulo de autenticación. En el caso de que el usuario operador abandone la página de monitorización o simplemente la cierre, el módulo de autenticación se encarga de cerrar la conexión con el *Topic*, que permanece establecida a lo largo de todo el período de actividad, para permitir la actualización del estado del sistema.

3.3.4.2. Módulo de visualización.

Como ya se ha comentado, el módulo de visualización es el encargado de la presentación de la información relevante sobre el estado de la centralita al usuario del sistema encargado de realizar la monitorización.

La monitorización de la centralita se realiza mediante una interfaz web que se ha elaborado empleado la tecnología *JSF* que simplifica el desarrollo de interfaces de usuario. *JSF* incluye, entre otras cosas, librerías que permiten construir una interfaz de usuario desde una página *JSP*, que es precisamente la tecnología que se ha empleado. A dicha página se le asoció un código script cuya función reside en refrescar la presentación de la información sobre el estado de la centralita. Es decir, la técnica seleccionada permite actualizar sólo las partes de la página que se han modificado, sin tener que recurrir a volver a representar la totalidad de la página.

Para construir la parte web del proyecto se hizo uso de las herramientas facilitadas por el entorno de programación empleado. Es decir, eclipse proporciona la posibilidad de crear distintos tipos de proyectos, cada uno con su estructura determinada, y una finalidad diferente. Uno de los posibles tipos de proyecto es el denominado "*Dynamic Web Project"* que permite la creación de páginas web empleando la tecnología *JSP*. Para poder manejar este tipo de proyectos fue necesaria la adhesión del *plugin* llamado *WebTools* que proporciona todo lo imprescindible para desarrollar una aplicación web.

El tipo de proyecto mencionado dispone de una estructura característica, siendo una de las carpetas más relevantes la denominada *WebContent*, que es la carpeta donde se almacenan los archivos *.jsp* y *.js* del proyecto.

3.3.4.2.1. Módulo de configuración del interfaz.

Para la construcción del módulo de configuración hizo falta la elaboración de una nueva página web cuya función reside en brindarle al usuario operador la posibilidad de personalizar su interfaz de trabajo. Para la realización de dicha página web se empleó la misma tecnología que en el caso del interfaz de operador y cuyas características ya se han descrito en apartados anteriores.

En la Figura 3.5 se representa el diagrama de flujo que describe el ciclo de vida del módulo de configuración del interfaz.

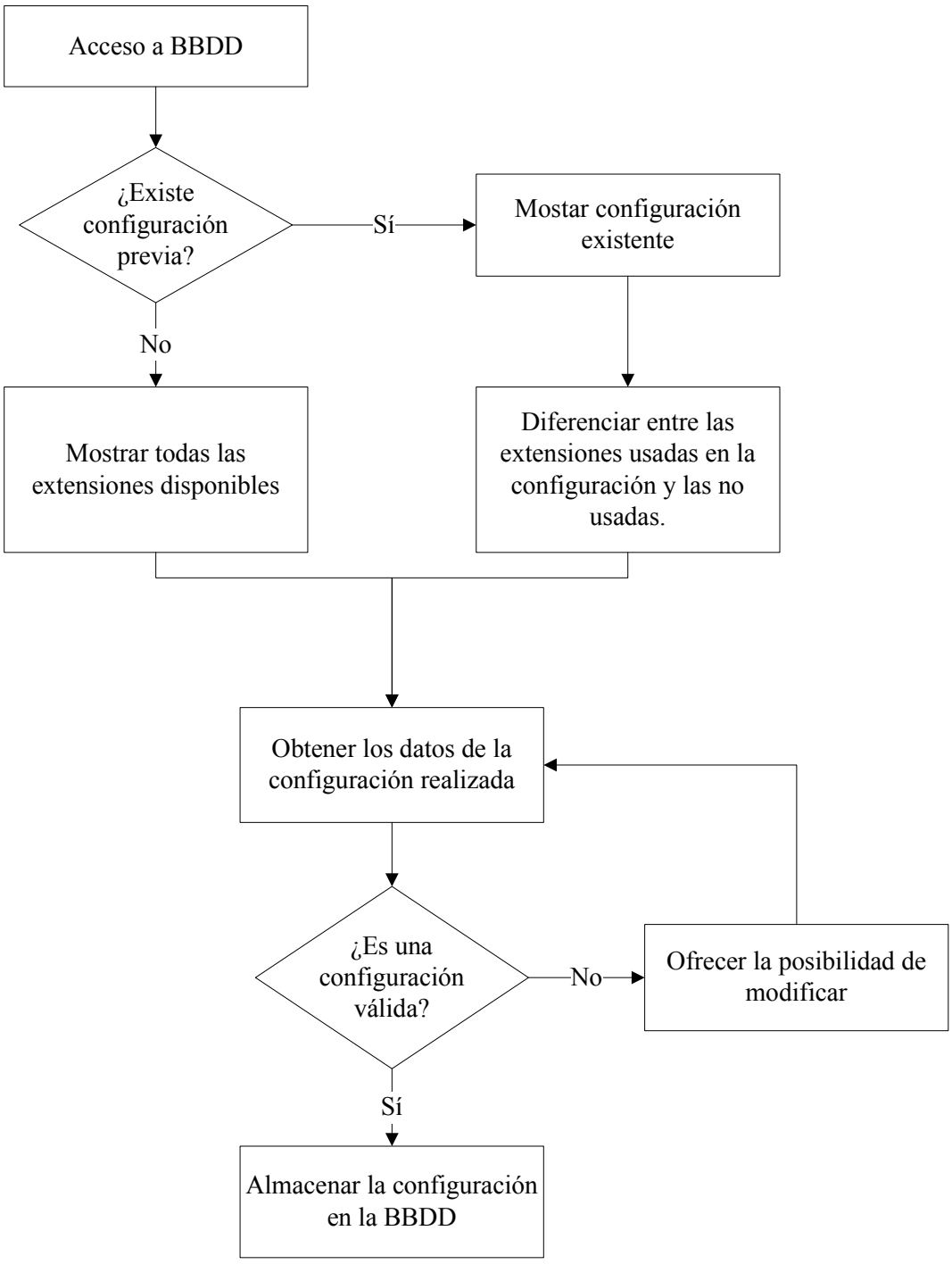

Figura 3.5: Diagrama de flujo del Módulo de Configuración del interfaz.

La primera acción llevada a cabo cuando el usuario accede a la página con la posible intención de realizar una configuración de su interfaz de trabajo, consiste en acceder a la base de datos para averiguar si dicho usuario ya había realizado alguna configuración previa.

En la Figura 3.6 se puede visualizar las tareas que se llevan a cabo en el momento de inicialización de la página.

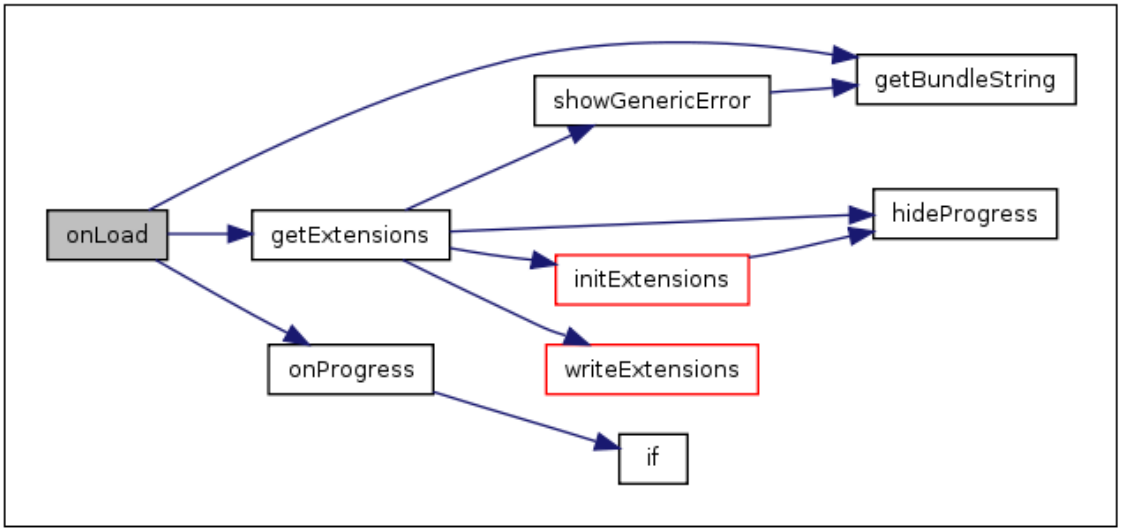

Figura 3.6: Grafo de llamada de la función de inicialización del Módulo de Configuración del interfaz.

En caso de existir una configuración previa, el módulo de configuración de interfaz le presenta al usuario los datos almacenados relativos a dicha configuración. En caso de que el usuario así lo solicite, se le ofrece la posibilidad de la elaboración de una nueva configuración mostrándole de manera diferenciada las extensiones ya configuradas y las disponibles en el sistema aún no empleadas en la configuración, dicha diferenciación se consigue a través de las funciones *initExtensions* y *writeExtensions* mostradas en la Figura 3.6. También se pueden observar registros correspondientes a las funciones auxiliares necesarias para lograr la funcionalidad descrita.

A continuación, el usuario puede plasmar sus preferencias sobre diferentes aspectos de la configuración del interfaz y una vez haya finalizado, le da la orden al módulo de configuración de registrar los datos relevantes. Tras recibir dicha orden, el módulo se encarga de comprobar que la configuración realizada sea válida y en caso de que no lo sea, se muestra un aviso al usuario explicando el error y se le ofrece la oportunidad de subsanarlo. Una vez enmendados los fallos, la configuración construida se envía para su almacenamiento a la base de datos.

La forma de realizar una configuración válida del terminal de operador se describe extensamente en el capítulo del manual de usuario pero se comentarán brevemente en este aparatado las posibilidades que ofrece la aplicación.

El usuario dispone de la posibilidad de seleccionar las extensiones del sistema que desea monitorizar y de distribuirlas en la pantalla de forma que considere más óptima para la realización de su tarea, dentro de la parrilla que permite colocar tres extensiones por fila. Puede realizar el traslado automático de las extensiones a la parrilla en el orden en el que aparecen en la lista o empleando la función que las ordena por su número. Si lo desea, también dispone de la posibilidad de realizar el traslado de forma manual, distribuyendo las extensiones de la forma que desee, con la única restricción de no dejar huecos libres en el medio. En cualquier momento puede añadir o eliminar extensiones de la configuración empleando los botones dispuestos a tal efecto. Otra funcionalidad consiste en la opción de elegir las subpantallas que desea que se le muestren durante la monitorización, es decir, existen cuatro configuraciones predeterminadas entre las cuales, el usuario puede seleccionar la que considere más idónea para él. También puede designar cómo desea que se identifiquen las extensiones en la parrilla de presentación, mediante su identificador *login* o mediante la inicial del nombre y el primer apellido del usuario.

En la Figura 3.7 y en la Figura 3.8 se visualizan los procesos llevados a cabo en el momento de traslado de la configuración realizada a la base de datos para su almacenamiento.

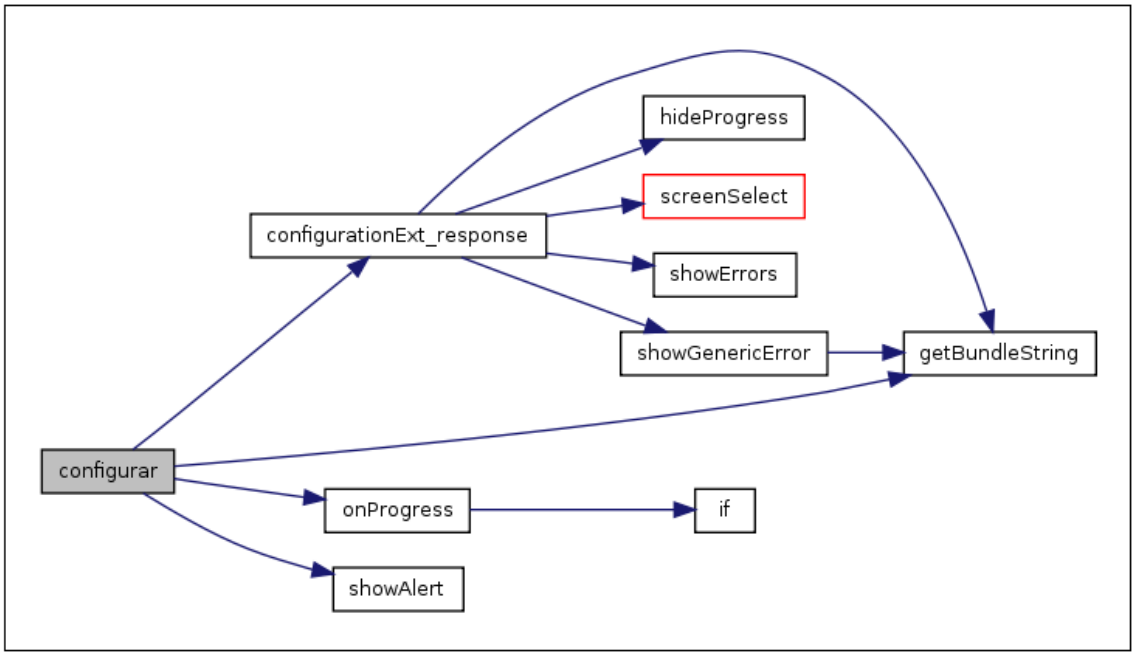

Figura 3.7: Grafo de llamada de la función *configurar*.

La función denominada *configurar* se encarga de almacenar en la base de datos la información referente a la parrilla elaborada por el usuario y tras comprobar que el proceso de almacenaje no ha producido ningún error, se puede observar en la Figura 3.7 que se invoca la función llamada *screenSelect* que es la encargada de guardar el resto de la información relevante sobre la configuración realizada.

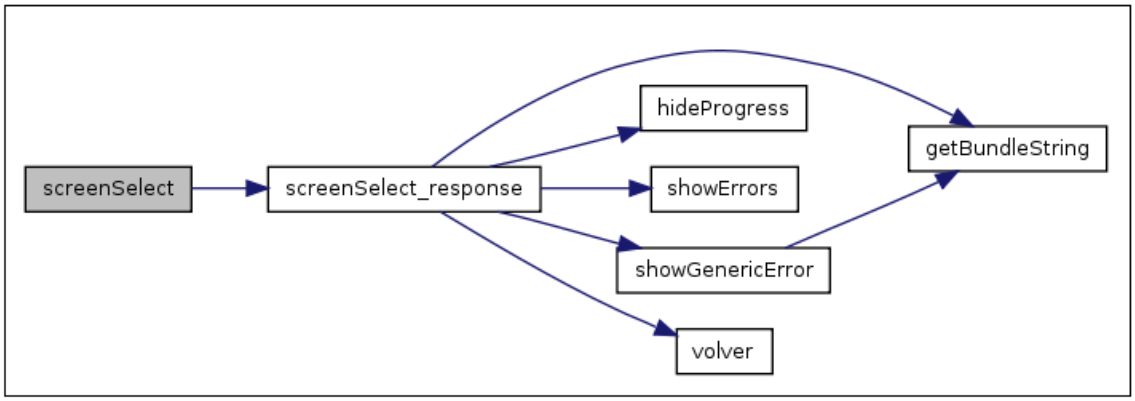

Figura 3.8: Grafo de llamada de la función *screenSelect*.

Al igual que en los otros módulos descritos, es importante aquí la presencia de numerosas funciones auxiliares para tratar los casos de error.

3.3.4.2.2. Módulo de control de contactos.

El módulo de control de contactos corresponde a otro de los componentes del módulo de visualización y como su nombre lo indica, se encarga de todo lo relacionado con la agenda del usuario operador. En la Figura 3.9 se puede apreciar el diagrama de flujo correspondiente a dicho módulo.

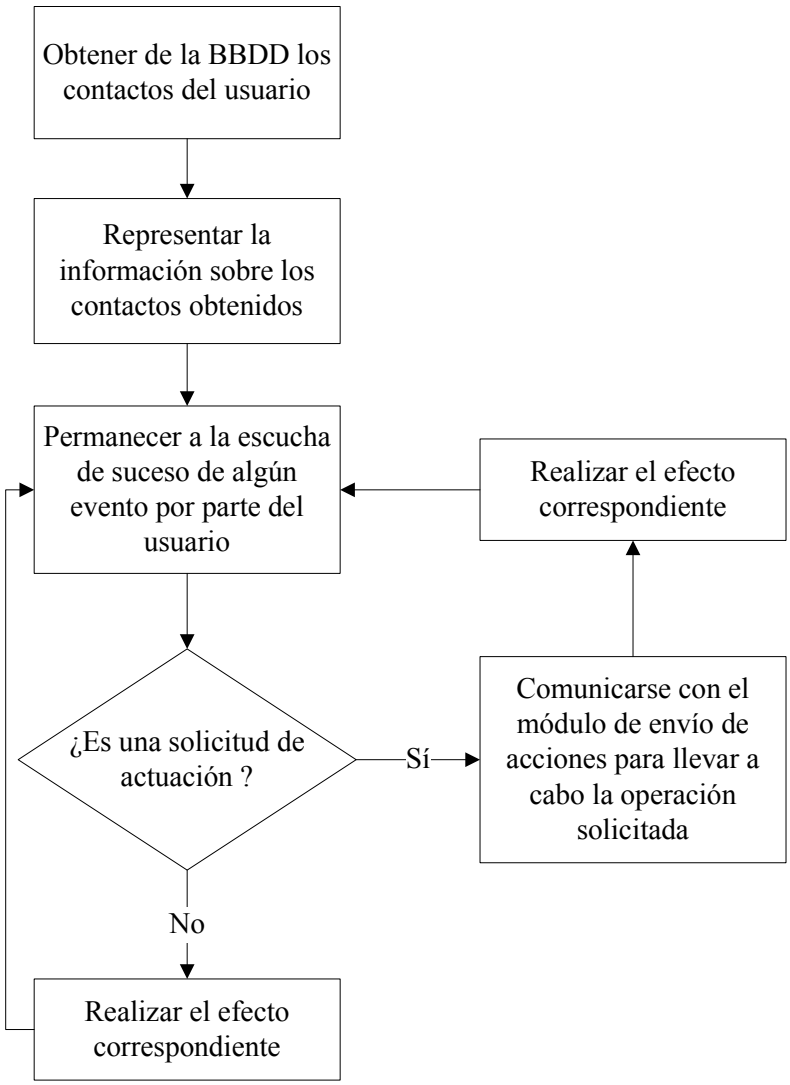

Figura 3.9: Diagrama de flujo del Módulo de Control de contactos.

En el momento en el que el usuario operador intenta acceder a la pestaña correspondiente al interfaz de operador, el presente módulo se encarga de comunicarse con el módulo externo del gestor de la base de datos, para obtener la información pertinente sobre los contactos que dicho usuario tiene almacenados en su agenda.

Una vez obtenidas las entradas de la agenda, el módulo de control de contactos realiza la representación de los mismos empleando las estructuras correspondientes definidas en el *css* empleado.

Tras finalizar la etapa de inicialización descrita, el presente módulo se encomienda a la escucha de eventos que pueda ocasionar el usuario, mediante el uso del ratón o teclado, sobre la superficie reservada a la representación de los contactos.

Los eventos mencionados pueden ser de dos tipos:

- Que impliquen comunicación con el módulo de envío de acciones.
- Que no impliquen dicha comunicación.

Los eventos pertenecientes al segundo grupo, son aquellos que simplemente corresponden a la solicitud de cambio de aspecto de los contactos. Cuando el usuario se posiciona con el ratón sobre una entrada de la agenda en concreto, ésta se representa con un

tamaño considerablemente aumentado para reflejar de forma inequívoca, cuál es el contacto implicado en la actividad del usuario.

En cambio, cuando el usuario genera un doble click, pulsa el botón de marcar o arrastra un número de teléfono hacia uno de los contactos, se hace necesaria la intervención del módulo de envío de acciones porque dichas situaciones corresponden a la solicitud del usuario de alterar la actividad de la centralita.

Si el usuario realiza un doble click sobre una entrada de la agenda, corresponde a la solicitud, por parte de éste, del establecimiento de una llamada entre el terminal telefónico del usuario operador y el número de teléfono correspondiente al contacto. Cuando ocurre el evento descrito, se ejecuta la función denominada *clickToCall* cuyo gráfico de llamada aparece en la Figura 3.10.

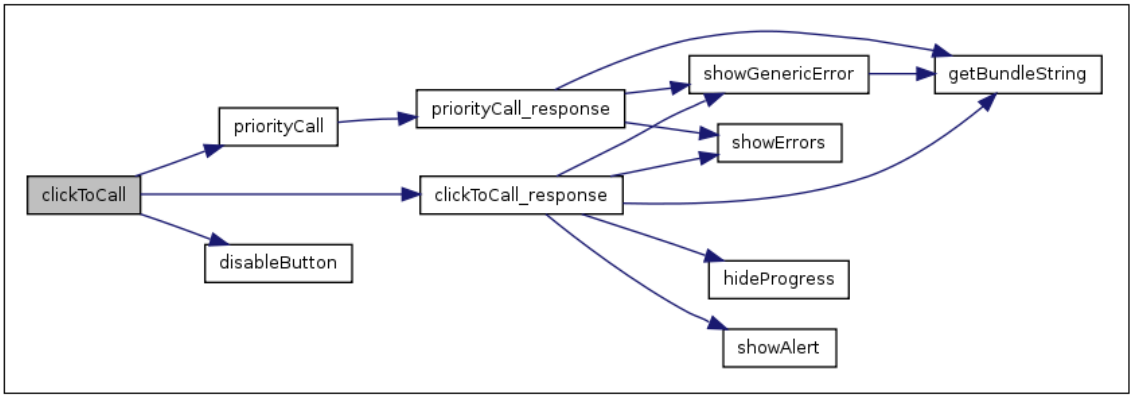

Figura 3.10: Grafo de llamada de la función *clickToCall*.

Como se puede observar en el gráfico de llamadas, para realizar el establecimiento deseado, previamente hay que comprobar si el usuario operador no se encuentra ya ocupado en una conversación. En caso de ser así, se hace necesario aparcar cualquier otra llamada establecida del usuario involucrado, ya que se supone que éste considera prioritario el establecimiento de la nueva llamada que acaba de solicitar. Para llevar a cabo dicha operación se realiza una comunicación con el módulo de envío de acciones para dirigir una orden hacia *Asterisk*. En esta ocasión se trata de la construcción y posterior envío de un objeto de la clase *ParkAction*, que es una de las acciones disponibles en la librería de *Asteriskjava*.

Una vez se hayan puesto en espera las posibles llamadas iniciadas del usuario, se realiza el establecimiento de la llamada solicitada. Para ello se acondiciona de manera adecuada un objeto de la clase *OriginateAction*, proporcionada también por la misma librería. Tras la inicialización de los atributos precisos, se establece una nueva conexión con el manager de *Asterisk*, se realiza el envió del objeto acondicionado y tras la recepción de la confirmación de la correcta ejecución, se cierra la conexión establecida.

Si por alguna razón no se consigue el correcto establecimiento de la llamada solicitada, se genera una excepción, que tras ser capturada, en la página web se visualizará como un mensaje informando del error ocurrido. Tratar esos posibles casos de error es el propósito que tienen las funciones auxiliares que aparecen en gráfico de llamada de la función *clickToCall*.

Otra operación importante que es posible realizar sobre los contactos de la agenda, es la transferencia de una llamada entrante o saliente.

Para provocar la transferencia descrita, el usuario operador debe seleccionar con el ratón el número de teléfono del interlocutor que desea conectar con un número registrado en la agenda de una de las llamadas del sistema, arrastrarlo hasta la zona donde se representa la agenda y soltar el número seleccionado sobre el contacto de la agenda deseado. La operación

descrita genera un evento, que gracias al empleo de la librería Script.aculo.us se puede capturar y el cual provoca la demanda de la intervención de la función denominada *tranferir*. En la Figura 3.11 se puede observar el gráfico de llamada de la mencionada función. Y a continuación, en la Figura 3.12 se observa el gráfico de las funciones que llaman dicha función. En dicho gráfico aparece una función denominada *write_Contact* que es la función que se encarga de generar la estructura con las entradas de la agenda que será presentada al usuario. Lo cual se explica con el hecho de que para conseguir que las entradas de la agenda sean sensibles a los eventos del tipo descrito más arriba, es necesario indicárselo en el momento de su creación.

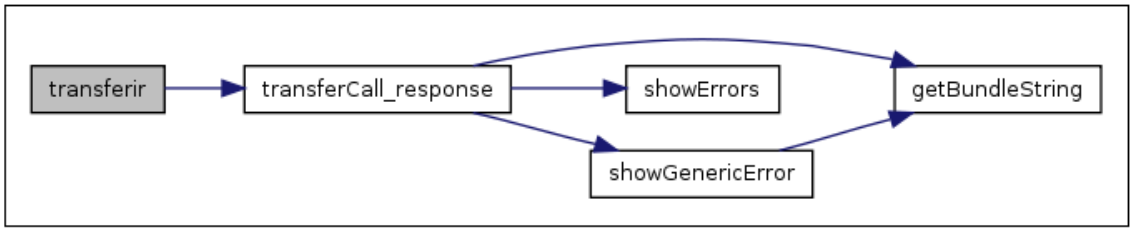

Figura 3.11: Grafo de llamada de la función *transferir*.

En la Figura 3.11 se puede observar que tras la llamada de la función *transferir*, intervienen las funciones auxiliares *showErrors* o *showGenericError* en el caso de que ocurra algún error durante la ejecución de la comunicación con el *bean* correspondiente.

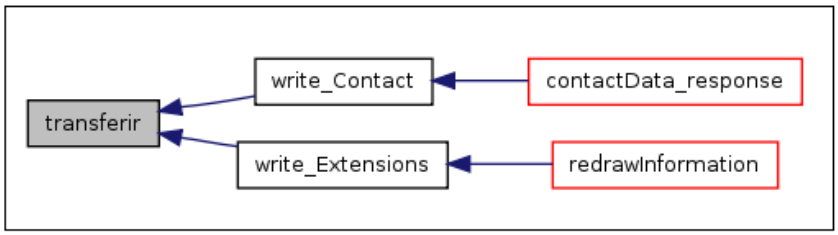

Figura 3.12: Grafo de llamantes de la función *trasferir*.

Como se puede apreciar en el diagrama de flujo del módulo de control de contactos, representado en la Figura 3.9, tras la realización de la comunicación con el módulo de envío de acciones, se lleva a cabo la representación del efecto visual que corresponda para facilitar el trabajo del usuario operador. En el caso, por ejemplo, de haber solicitado un establecimiento de llamada con un registro de la agenda, tras su ejecución, el contacto involucrado permanecerá parpadeando unos segundos para que el usuario que se está encargado de la monitorización del sistema pueda asegurarse de que el establecimiento de la llamada se ha realizado con el número correcto.

Otra funcionalidad implementada por el presente módulo consiste en permitir al usuario buscar registros entre las entradas disponibles en su agenda. En la subpantalla reservada para la representación de los contactos del usuario existe un espacio preparado para la recepción de texto introducido por el usuario. Cuando el usuario escribe algo en dicho espacio reservado, el módulo de control de contactos se encarga de recoger los caracteres introducidos y comunicarse con el módulo externo de la base de datos para buscar contactos que respondan al patrón deseado. Una vez obtenidas las entradas que cumplen con el patrón son devueltas al módulo de control de contactos para su representación en la pantalla del usuario. El usuario puede realizar todas las búsquedas que desee y si quiere volver a la representación de todas las entradas disponibles en su agenda, tan sólo tiene que borrar lo que había introducido en el espacio reservado para el patrón de búsqueda.

Por último, el usuario dispone de la posibilidad de realizar el establecimiento de una llamada, a través del interfaz, sin la necesidad de que éste se encuentre registrado dentro de la agenda. Para ello, basta con introducir el número destinatario deseado en el campo descrito arriba y pulsar el botón denominado *Marcar*. Como en otras ocasiones, dicha acción provocará que el terminal telefónico del usuario operador empiece a sonar y en cuanto sea descolgado, comenzará el intento de establecimiento de llamada hacia el número introducido.

3.3.4.2.3. Módulo de control de llamadas.

En la Figura 3.13 se representa el diagrama de flujo del módulo de control de llamadas, que como ya se ha explicado, es uno de los componentes del módulo de visualización. El presente mecanismo se encarga de dos grandes tareas bien diferenciadas. En una primera instancia, es el responsable de presentar al usuario el estado actualizado en todo momento de las llamadas entrantes y salientes del sistema. A su vez, es el encargado de detectar las posibles acciones que el usuario operador pueda realizar sobre las llamadas existentes y de ponerse en contacto con el módulo de envío de acciones para que éste le envíe la orden pertinente al manager de *Asterisk*.

La primera gran intervención del módulo descrito en esta sección tiene lugar en el momento del acceso del usuario a la aplicación. Como es lógico, se hace necesaria una consulta con el *Asterisk* para obtener la información pertinente sobre las llamadas existentes en el sistema en ese instante. Para conseguir la información precisada, el módulo de control de llamadas se pone en contacto con el módulo de envío de acciones y le transmite la petición de la demanda de información sobre el estado actual.

Cuando el módulo de envío de acciones recibe dicha petición, de nuevo hace uso de la librería *Asteriskjava* y construye un objeto de tipo *StatusAction*. Establece la conexión con el manager de *Asterisk* y le envía la acción construida empleando la función *sendEventGeneratingAction*. La mencionada función se emplea cuando se envían acciones que provocan la generación de eventos por parte de la centralita. Mediante dicho procedimiento, la respuesta que se recibe por parte de *Asterisk* es una colección de eventos de tipo *StatusEvent*, más concretamente, se genera un evento de este tipo por cada uno de los canales activos en el sistema en el momento de la consulta.

La responsabilidad de analizar la repercusión de los eventos recibidos recae en el módulo de envío de acciones. Es decir, éste se encarga de guardar la información recibida en una estructura tipo *Iterator* y se dispone a examinar cada uno de los eventos para obtener información sobre las llamadas que ya se encontraban establecidas en el momento en el que el usuario operador accede a la página del interfaz. Del análisis de la estructura, el módulo es capaz de establecer el origen y destino de cada llamada, tanto a nivel de usuario como con respecto al canal que está usando cada uno. También es capaz de determinar si la llamada se encuentra retenida en el momento del acceso o en cambio es la que ocupa al usuario en ese instante. Y por último, establece el tiempo que ha transcurrido desde el inicio de la cada llamada obtenida.

Organiza todos los datos obtenidos para volver a enviárselos al módulo de control de llamadas para su representación en la pantalla del usuario. Para terminar, no hay que olvidar cerrar la conexión con el manager de *Asterisk*.

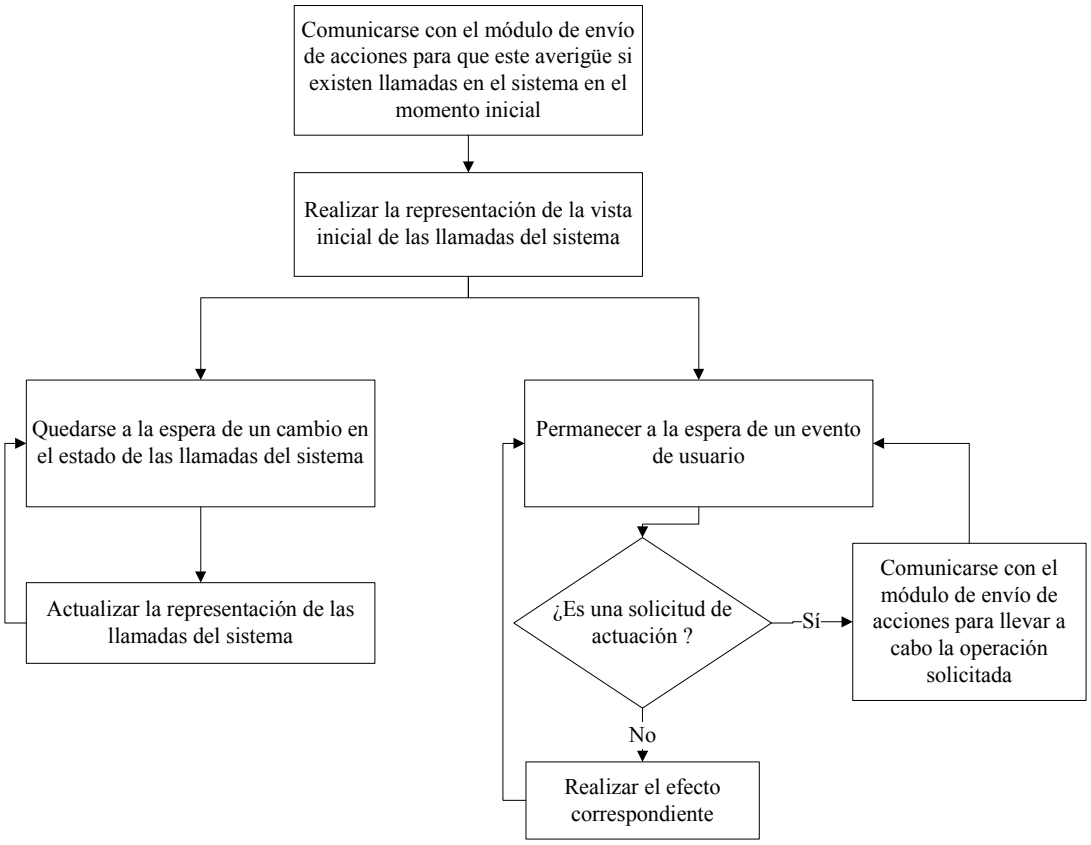

Figura 3.13: Diagrama de flujo del Módulo de Control de llamadas.

Cuando el módulo de control de llamadas recibe la respuesta con la información sobre el estado actual del sistema a su petición inicial de información, se encarga de construir las estructuras pertinentes que representan las llamadas del sistema, para ello emplea las funciones *write_PhoneIn* y *write_PhoneOut* cuyos diagramas de llamada se muestran en la Figura 3.14 y en la Figura 3.15 respectivamente.

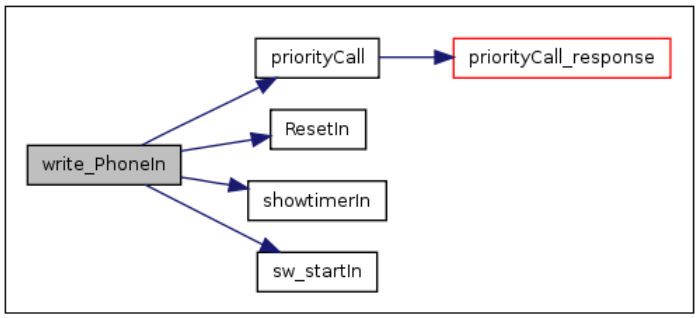

Figura 3.14: Grafo de llamada de la función *write_PhoneIn*.

En esta primera representación se puede observar como en el momento de la construcción de la estructura correspondiente a una llamada entrante, se emplean funciones encargadas de inicializar el estado del contador que se encargará de presentarle al usuario operador la información sobre la duración de la presente llamada a partir del momento en que ésta se inicie. También, mediante el uso de la función *priorityCall* se le asigna a la llamada la posibilidad de ser priorizada con respecto a las otras posibles llamadas hacia el mismo usuario. Es decir, dicha funcionalidad permite al usuario escoger en todo momento la llamada que desea

atender sin importar el orden en el que se hayan originado. Mediante un simple doble click con el ratón sobre la llamada deseada, el terminal telefónico del operador se comunica con el interlocutor de la llamada seleccionada, mientras que, en el caso de existir otras llamadas entrantes hacia el mismo usuario, éstas quedan aparcadas a la espera de ser recuperadas.

La aplicación es capaz de provocar el efecto descrito gracias a la parte del módulo de control de llamadas que permanece a la escucha de los eventos provocados por el usuario. En el momento de detectar un evento, analiza su tipo y en función de eso, desencadena un procedimiento u otro. Sin embargo, la estructura seguida siempre es la misma, es decir, el módulo de control de llamadas construye una petición para el módulo de envío de acciones y éste es el encargado de transmitir la petición a *Asterisk* haciendo uso de nuevo de las funcionalidades proporcionadas por la librería.

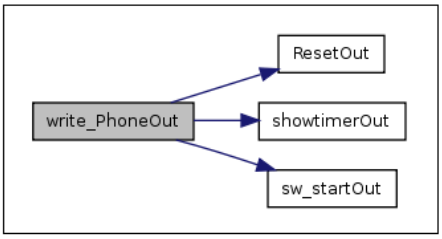

Figura 3.15: Grafo de llamada de la función *write_PhoneOut*.

En el caso de esta segunda ilustración, se observa que para las llamadas salientes del sistema también se realiza la inicialización del futuro contador de la duración de la llamada pero no existe la posibilidad de priorizarla ya que carece de sentido en esta situación.

Para conseguir disponer en todo momento de la información actualizada sobre el estado de las llamadas del sistema, el módulo, tras la representación de la situación inicial, se queda a la espera de que el módulo de procesado de eventos lo avise de que se ha producido un cambio en el estado de las llamadas. En dicho instante, tras el procesado de la información recibida, el módulo se encarga de la actualización de la representación de las llamadas entrantes y salientes.

3.3.4.2.4. Módulo de control de estado.

Como ya se ha comentado en el apartado anterior, el módulo de control de estado está compuesto por dos grandes bloques cuya especificación se detalla a continuación.

3.3.4.2.4.1. Estado de las extensiones.

La función del módulo de estado de las extensiones reside en la capacidad de presentar en todo momento cualquier cambio en la situación de las extensiones monitorizadas del sistema. A su vez, recae sobre éste la responsabilidad de absorber las peticiones de interacción del usuario operador referentes a las extensiones.

En la Figura 3.16 se puede visualizar el diagrama de flujo que representa las dos tareas principales del módulo encargado del estado de las extensiones.

En el momento de inicio de la actividad, el presente módulo establece una comunicación con el módulo de envío de acciones para solicitarle que realice una consulta sobre el estado de las extensiones.

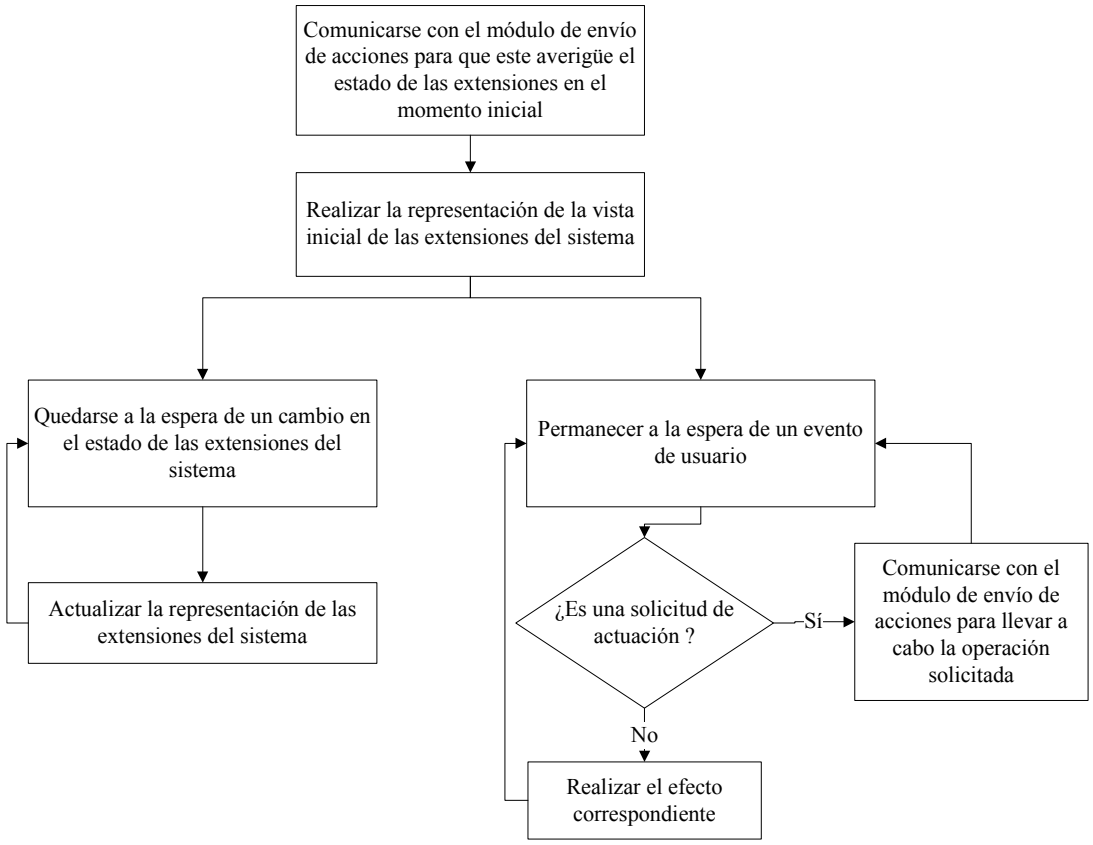

Figura 3.16: Diagrama de flujo de Módulo de Estado de las extensiones.

El módulo de envío de acciones ante dicha petición, establece una conexión con el manager de *Asterisk* para satisfacer la demanda recibida. En esta ocasión, a partir de la información disponible sobre las extensiones monitorizadas, se realiza la consulta del estado de los canales correspondientes. Disponiendo de dicha información es posible averiguar si una extensión está libre o se encuentra ocupada en una conversación, obteniendo también el interlocutor de la misma. Otras opciones posibles son que en el momento del acceso a la monitorización, una extensión se encuentre sonando ya que tiene una llamada entrante o interna todavía no atendida, o simplemente se encuentre no disponible por algún problema en el registro del terminal telefónico asociado al usuario afectado.

Tras obtener toda la información relevante sobre el estado de las extensiones, el módulo de envío de acciones se la encamina hacia el módulo de estado de las extensiones. Éste a su vez, construye cada una de las extensiones mediante el uso de la función *write_Extensions* cuyo diagrama de llamada se muestra en la Figura 3.17.

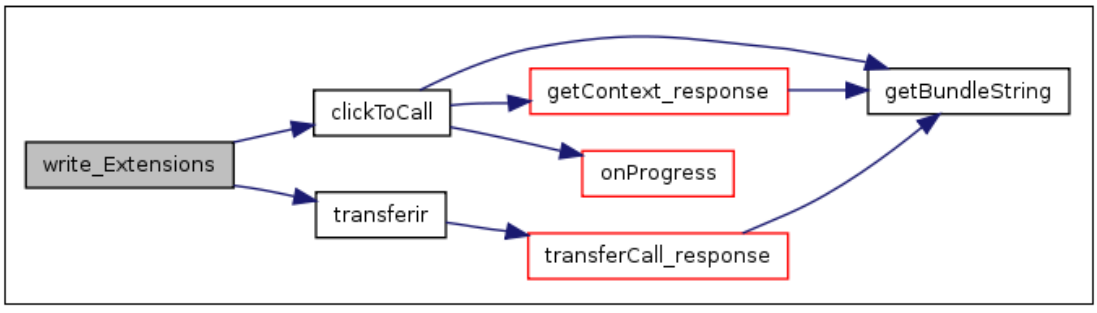

Figura 3.17: Grafo de llamada de la función *write_Extensions*.

Dicho diagrama muestra que cada extensión, al igual que ocurría en el caso de contactos, se construye de forma que permitirá el establecimiento de una llamada con esa extensión como destino y admitirá que las llamadas del sistema puedan ser transferidas hacia ésta.

En el momento de la construcción de la extensión, se establece que será sensible al evento generado por el usuario de doble click, de forma que cuando éste se produzca, la aplicación recurrirá a los servicios de la función *clickToCall* para establecer una llamada entre el usuario operador y la extensión seleccionada. En este caso también será necesario primero aparcar las posibles llamadas entrantes al sistema que tengan al usuario operador como destinatario. Para establecer una conversación con una de las extensiones del sistema será preciso averiguar en qué contexto se encuentra trabajando ésta, es por ello que aparece en el diagrama la función *getContext_response*, a la vez que otras funciones auxiliares.

Al igual que en el caso de los contactos de la agenda, las extensiones monitorizadas podrán ser los nuevos destinatarios de una llamada entrante o saliente del sistema, es decir, el usuario operador dispone de la posibilidad de transferir las llamadas del sistema hacia cualquiera de las extensiones simplemente seleccionando el número en el cuadrante reservado a las llamadas y arrasándolo hacia la extensión destinataria deseada. Dicha acción provocará una transferencia ciega, es decir, la conversación entre los anteriores interlocutores, en caso de haber sido ya iniciada, se interrumpirá, el terminal telefónico del usuario de la extensión destino comenzará a indicar la entrada de una nueva llamada, mientras que al interlocutor seleccionado en la transferencia se le reproducirá música en espera hasta que el otro descuelgue.

Otra labor importante del módulo de estado de las extensiones reside en permanecer a la espera de los cambios en el sistema que anuncia el módulo de procesado de eventos. Cualquier alteración en la centralita que afecte al estado de las extensiones monitorizadas es procesada y comunicada al módulo que nos ocupa. Éste a su vez, es el encargado de realizar la actualización del estado del sistema que se presenta al usuario.

3.3.4.2.4.2. Estado de las líneas.

La función del módulo de estado de las líneas responde a un diagrama de flujo muy similar al de los dos últimos módulos, como se puede observar en la Figura 3.18 presentada a continuación.

En dicha ilustración se puede advertir que este módulo también tiene dos funciones bien diferenciadas, se encarga de permanecer a la espera de un cambio en el estado de las líneas, a la vez que de mantenerse a la escucha de un posible evento ocasionado por el usuario.

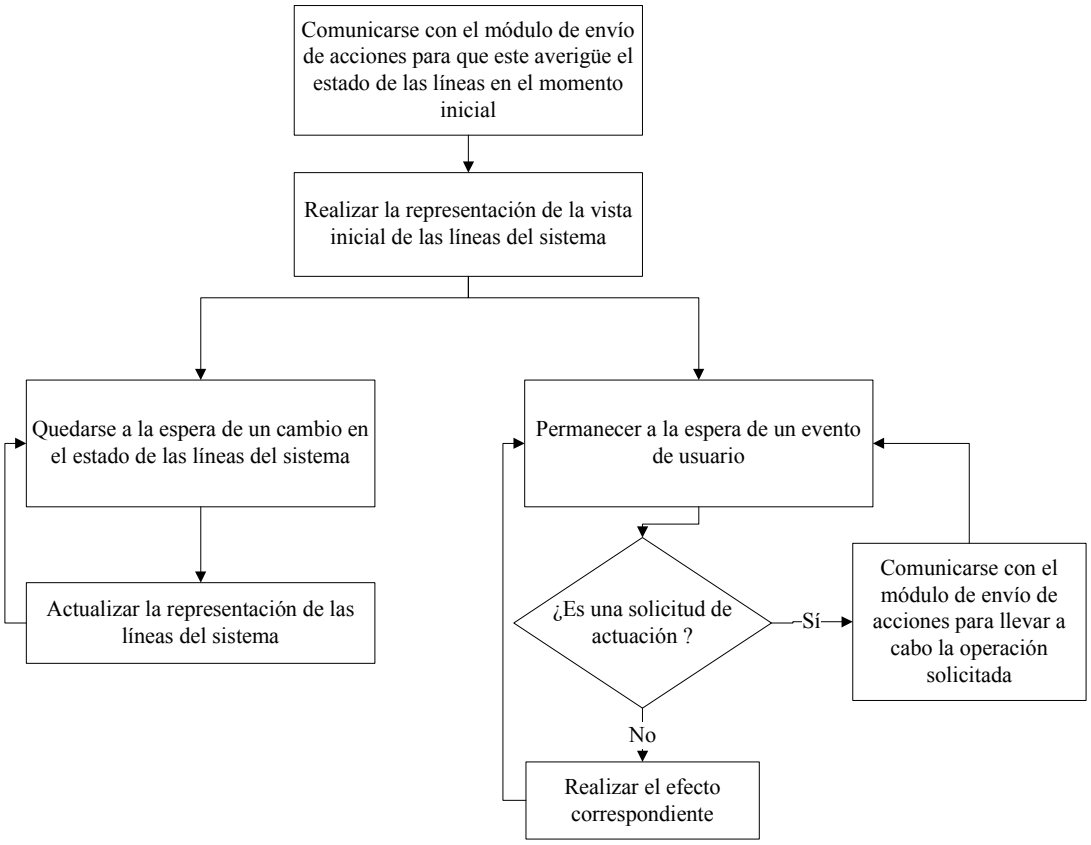

Figura 3.18: Diagrama de flujo del Módulo de Estado de las líneas.

Como en los demás módulos, en la etapa de inicialización, lo primero es solicitar al módulo de envío de las acciones información sobre el estado en el momento actual. Cada línea que se representa en la pantalla del usuario operador corresponde a un canal, con lo cual, cuando el módulo de envío de acciones recibe la respuesta de *Asterisk* a su petición de información sobre el estado, puede relacionar los canales disponibles, los que no lo están y los que se encuentran ocupados y le proporciona dicha información al módulo de estado de las líneas.

Para realizar la representación de las líneas a partir de la información obtenida, se emplea la función denominada *write_Port* cuyo gráfico de llamada aparece representado en la Figura 3.19.

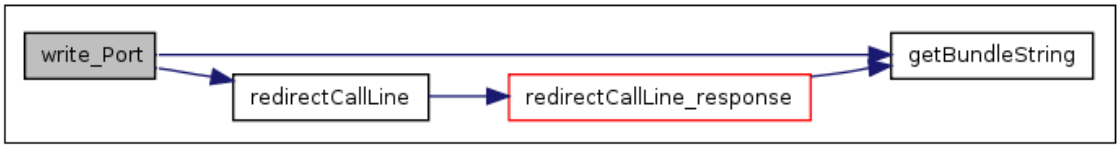

Figura 3.19: Grafo de llamada de la función *write_Port*.

Dicha función se encarga de representar las líneas configuradas en el sistema mediante un esquema de *RJ45* (*Registered Jack 45*) que en caso de encontrarse realmente operativas aparecerán con un pequeño led verde y en caso de estar desconectadas de color gris. En el momento en que el canal correspondiente a una línea se encuentre en uso, ésta se representará en color rojo y cuando se libere volverá a su estado de reposo.

Del grafo de llamada de la función *write_Port* se puede deducir que ésta usa una función denominada *redirectCallLine* cuya tarea consiste en permitir el establecimiento de una llamada por una línea en concreto, es decir, una línea que elija el usuario operador.

Cuando el usuario encargado de la monitorización desee establecer una llamada a través de una línea en concreto, debe seleccionar un número de teléfono entre los disponibles en su agenda o teclear uno en el espacio reservado a tal efecto y arrastrarlo, con la ayuda del ratón hacia la línea de salida que desee emplear.

El módulo de estado de las líneas detecta el intento de encaminamiento descrito y reacciona mostrándole al usuario la línea preseleccionada en color naranja para que éste pueda estar seguro en todo momento de que no se equivoca.

Una vez el usuario escoge definitivamente la línea deseada, no tiene más que soltar el número que estaba arrastrando sobre dicha línea. El módulo de estado de las líneas, detecta el evento descrito y reacciona invocando la función *redirectCallLine* cuyo grafo de llamada se representa en la Figura 3.20.

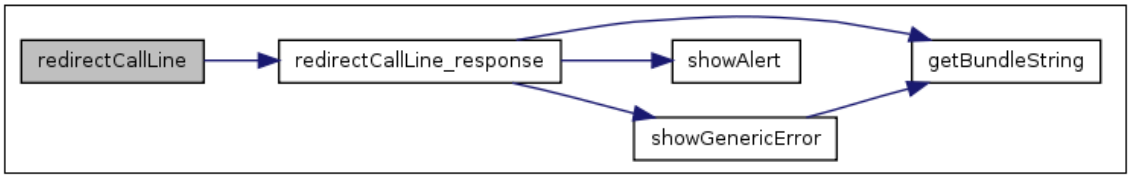

Figura 3.20: Grafo de llamada de la función *redirectCallLine*.

Para llevar a cabo el encaminamiento deseado, el módulo de estado de líneas se pone en contacto con el módulo de envío de acciones notificándole la orden generada por el usuario.

El módulo de envío de acciones recurre una vez más a las funcionalidades proporcionadas por la librería *Asteriskjava*. Establece una conexión con el manager de *Asterisk* y crea un objeto de la clase *OriginateAction* e indica que lo que se desea en este caso es originar una llamada y por lo tanto, la aplicación a utilizar será la denominada *Dial*. Se dedica a inicializar los atributos correspondientes al origen de la llamada, que será el usuario que ha originado la petición, el contexto al que pertenece, la línea que ha sido seleccionada para el encaminamiento y el número de teléfono con el que se desea establecer la comunicación.

El manager de *Asterisk* envía una respuesta indicando el éxito de la operación en caso de que todo haya ido según lo previsto. Si no es así, se envía un mensaje de error y se informa al usuario de que ha sido imposible establecer la comunicación. En caso de que la centralita haya procesado la petición de forma satisfactoria, comenzará a sonar el terminal telefónico del usuario operador y cuando éste descuelgue, se iniciará el establecimiento de la llamada solicitada a través de la línea escogida a tal efecto.

La segunda función importante del módulo de estado de las líneas reside en permanecer a la espera de las actualizaciones de estado enviadas por el módulo de procesado de eventos. En el momento de recibir el aviso de cambio en el estado de las líneas del sistema, el módulo se encarga de actualizar la representación gráfica para avisar al usuario de las alteraciones acontecidas.

3.3.4.3. Módulo de procesado de eventos.

En la Figura 3.21 se presenta el diagrama de flujo del módulo de procesado de eventos.

Como ya se he mencionado en la descripción de la arquitectura de la aplicación, el presente módulo está formado a su vez por varios componentes. Su primera función clave reside en la implementación de la tarea de permanecer a la escucha de los eventos generados por *Asterisk*.

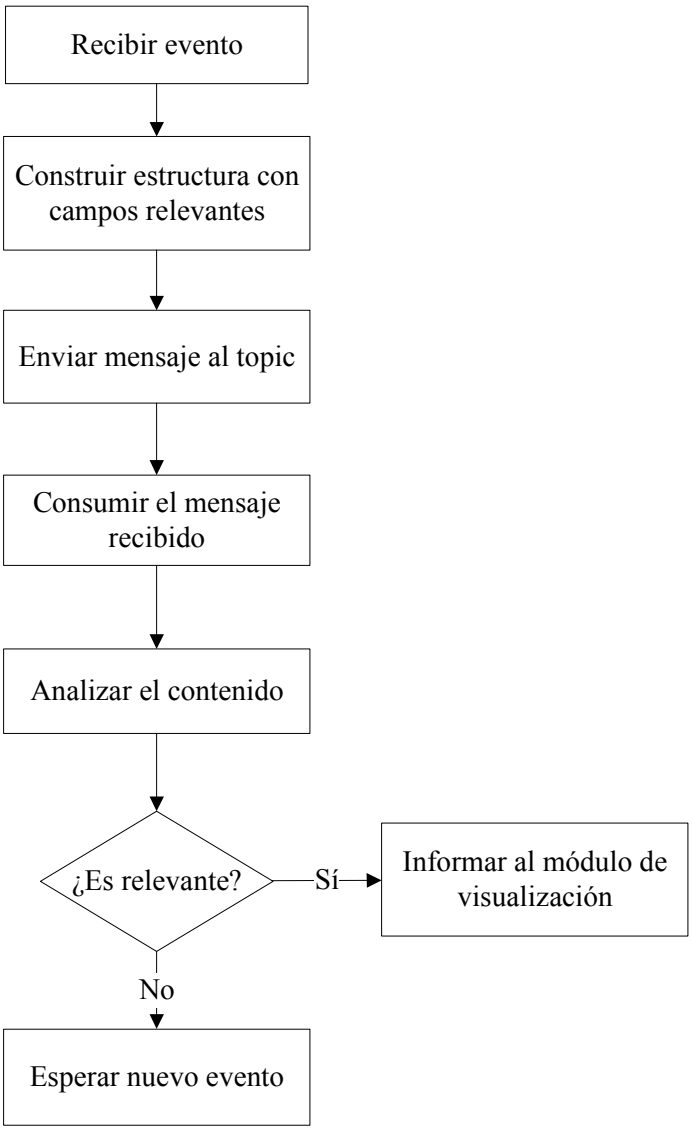

Figura 3.21: Diagrama de flujo del Módulo de Procesado de eventos.

Para conseguir la implementación de esta parte del módulo, se aprovecharon las prestaciones del *Jboss*, el cual describe en el apartado de las herramientas. Dicho servidor de aplicaciones ofrece la posibilidad de construir un *MBeans* de servicio que permite añadirle funcionalidad al servidor.

La idea reside en desplegar en el servidor un archivo tipo *SAR* (*Service ARchive*) que permite la creación de un servicio independiente de otros componentes del proyecto.

Con la finalidad de construir dicho servicio hubo que crear un nuevo subproyecto dentro del espacio de trabajo del Eclipse. Dicho proyecto debía cumplir con una estructura determinada para lograr la funcionalidad deseada.

Para conseguir identificarlo como un servicio dentro del servidor, una de las reglas que se debían cumplir residía en la creación de una carpeta llamada META-INF que debía contener un archivo denominado *jboss-service.xml* con la distribución mostrada en la Figura 3.22:

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <mbean code="services.ServAsteriskEventsService"

    name="services:service=ServAsteriskEventsService">

    </mbean>
    <loader-repository>
           services:service=ServAsteriskEventsService.sar
    </loader-repository>
</server>
```

Figura 3.22: Representación del archivo *jboss-service.xml.*

Adicionalmente, el proyecto en cuestión debía disponer de un paquete que contuviera dos clases con unas características muy determinadas. La primera de ellas debía ser la declaración de una sencilla interfaz que debía heredar de la clase *ServiceMBean* y presentar la declaración de los métodos pertinentes.

La segunda componente era una clase que debía implementar la interfaz anterior y heredar de la clase *ServiceMBeanSupport*, y consecuentemente proporcionarle una implementación a los métodos definidos en la interfaz.

Era importante tener en cuenta que los nombres de esas clases debían cumplir también ciertas reglas. Como el servicio se denominó *ServAsteriskEventsService*, la interfaz descrita antes debía comenzar así su nombre añadiéndole el sufijo de *MBean*. Mientras que la clase que realizaba la implementación de dicha interfaz debía adquirir el mismo nombre que el servicio.

A partir de este punto, el proyecto creado ya cumplía con los requisitos necesarios para que el servidor lo identificara claramente como un servicio, sin embargo todavía faltaba dotarlo de la funcionalidad deseada.

Para ello se tuvieron que añadir otros dos paquetes, uno denominado *manager* que contenía la implementación de la conexión con el Manager de *Asterisk*, y otro denominado *action* para proporcionar la posibilidad de envío de acciones a la *PBX*.

El paquete *manager* contenía las clases java que permitían la implementación de la conexión con la centralita *Asterisk*. Una parte era la encargada de controlar el ciclo de vida del servicio y la otra de la comunicación propiamente dicha con el *manager*.

La parte de la comunicación con el *manager* se realizaba mediante un objeto de la clase denominada *AsteriskManager*, cuya característica principal reside en que debía heredar de la clase *Thread* e implementar la interfaz *ManagerEventListener*. De esta forma se conseguía que el programa fuera un hilo de ejecución independiente y que fuera capaz de recibir los eventos generados por el manager de *Asterisk*. Su funcionamiento se basa en la implementación del método llamado *onManagerEvent* declarado en la interfaz y cuya ejecución tiene lugar cada vez que se recibe un nuevo evento.

Una vez recibido el evento y realizado su primer análisis para la extracción de los campos que proporcionan la información relevante, entra en juego el segundo componente del módulo estudiado. En esta ocasión se trata del modelo de paso de mensajes que se emplea para conseguir enviar el resumen del evento a la clase java que se encarga de averiguar la repercusión de éste en el estado del sistema. Como se explicó en el apartado de herramientas utilizadas, para dicha labor se empleó la tecnología *JMS*.

Como la aplicación se diseñó con la idea de que pudieran existir varios usuarios operadores monitorizando el estado del sistema, el paso de mensajes debía permitir que los mensajes generados por una única fuente pudieran ser recibidos por varios consumidores. Con la tecnología *JMS* se consigue la situación descrita construyendo un *topic*, cuya funcionalidad es precisamente la buscada.

Cada vez que se finaliza el análisis preliminar del evento recibido, se crea una nueva conexión con el *topic* empleado y se envía dicha información hacia sus posibles consumidores interesados.

Se llega así hasta el último componente del módulo de procesado de eventos, el cual se encarga realmente de decidir cuál es la relevancia del evento recibido para la actualización del estado del sistema presentado al usuario.

Este último componente está implementado mediante una clase java que mediante el empleo de la tecnología *JMS* es capaz de recibir los mensajes enviados por el servicio descrito antes. La mencionada clase se denomina *OperatorManagementBean* e implementa la interfaz *MessageListener* para permitir que se convierta en el receptor de los mensajes enviados por parte del componente anterior. Cada vez que se crea un nuevo objeto de esta clase, se estable una nueva conexión de paso de mensajes con el servicio emisor. En el momento que se recibe un nuevo mensaje, se invoca el método denominado *onMessage* que define el plan de actuación a partir de este punto, como se puede observar en la Figura 3.23.

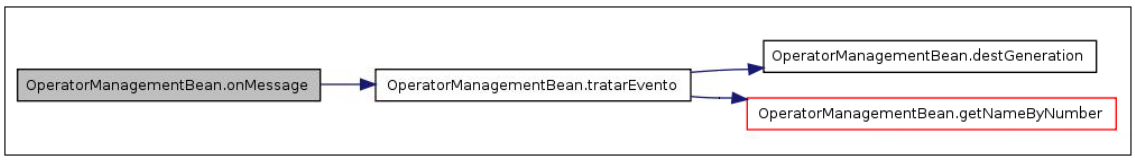

Figura 3.23: Grafo de llamada de la función *onMessage*.

Por cada posible usuario que se encuentre monitorizando el sistema, se crea una nueva instancia de dicha clase, por lo tanto, cada una de ellas podrá acceder a la información y sacar las conclusiones pertinentes con respecto a la relevancia en función de la parte del sistema que se encuentre monitorizando.

El método invocado por *onMessage* es el que implementa la funcionalidad del análisis del evento recibido para obtener la información actualizada del estado de la centralita y se denomina *tratarEvento*. Se examina cada evento, en función del tipo que sea, de forma diferente y se decide qué indica realmente con respecto a la situación del sistema. Si el evento recibido realmente resulta ser relevante, a continuación se construyen estructuras con información actualizada sobre las llamadas del sistema, el estado de las extensiones, el estado de las líneas y por último, se devuelven como respuesta a las peticiones de los diferentes módulos encargados de la visualización de los datos descritos.

La relación que existe entre los módulos de visualización y el módulo de procesado de eventos es la siguiente. Cada uno de los módulos de visualización, tras representación de la situación inicial del sistema, se encarga de realizar una petición de información actualizada sobre los componentes que los ocupan al módulo de procesado de eventos y se quedan a la espera de su respuesta. El componente encargado del análisis de los eventos, se encuentra a la escucha permanente de éstos y cuando considera que ha ocurrido un cambio en el estado de la centralita, envía la información actualizada hacia el módulo correspondiente. La situación descrita se puede apreciar en la Figura 3.24 donde se representa el diagrama de llamada de la función invocada como petición de situación actualizada, denominada *comprobarRelevante*. Se advierte en la imagen, que una vez recibida la respuesta, se ejecuta la función *redrawInformation* cuyo diagrama de llamada aparece en la Figura 3.25.

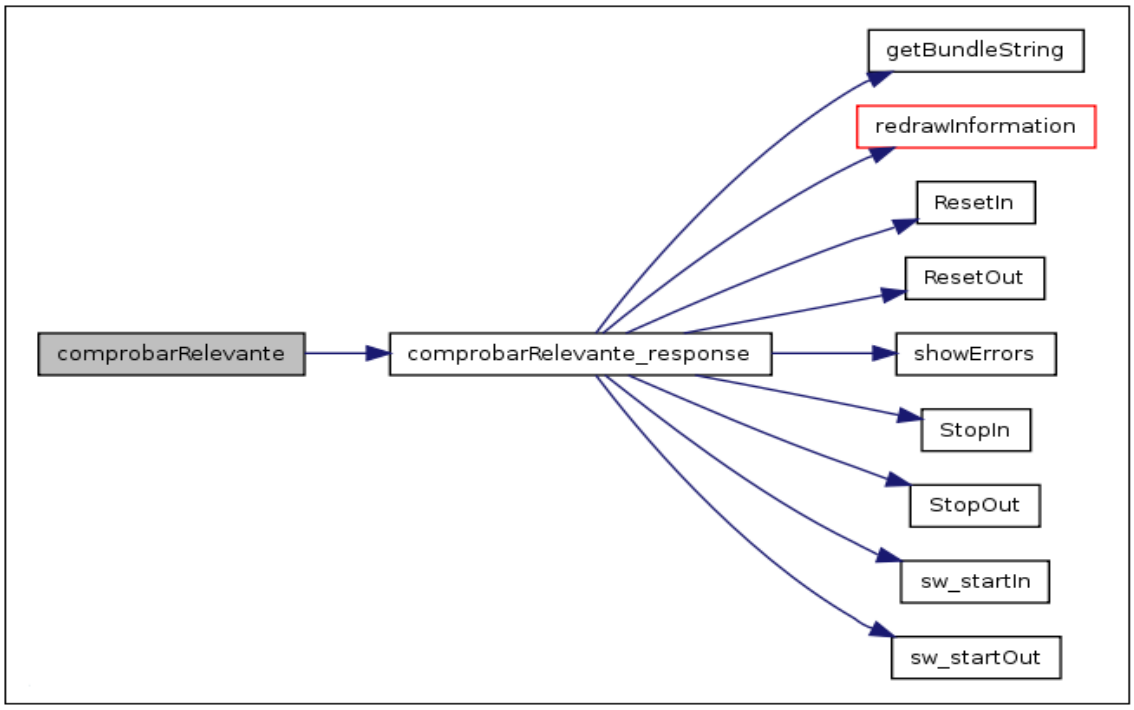

Figura 3.24: Grafo de llamada de la función *comprobarRelevante*.

El mencionado diagrama describe la reacción de cada uno de los módulos de visualización ante la recepción de la información actualizada del estado del sistema. Cada uno de ellos se encarga de la representación al usuario de la parte de información que le corresponde como ya se ha explicado para cada uno de los casos en los apartados anteriores y se puede apreciar como resumen en la Figura 3.25.

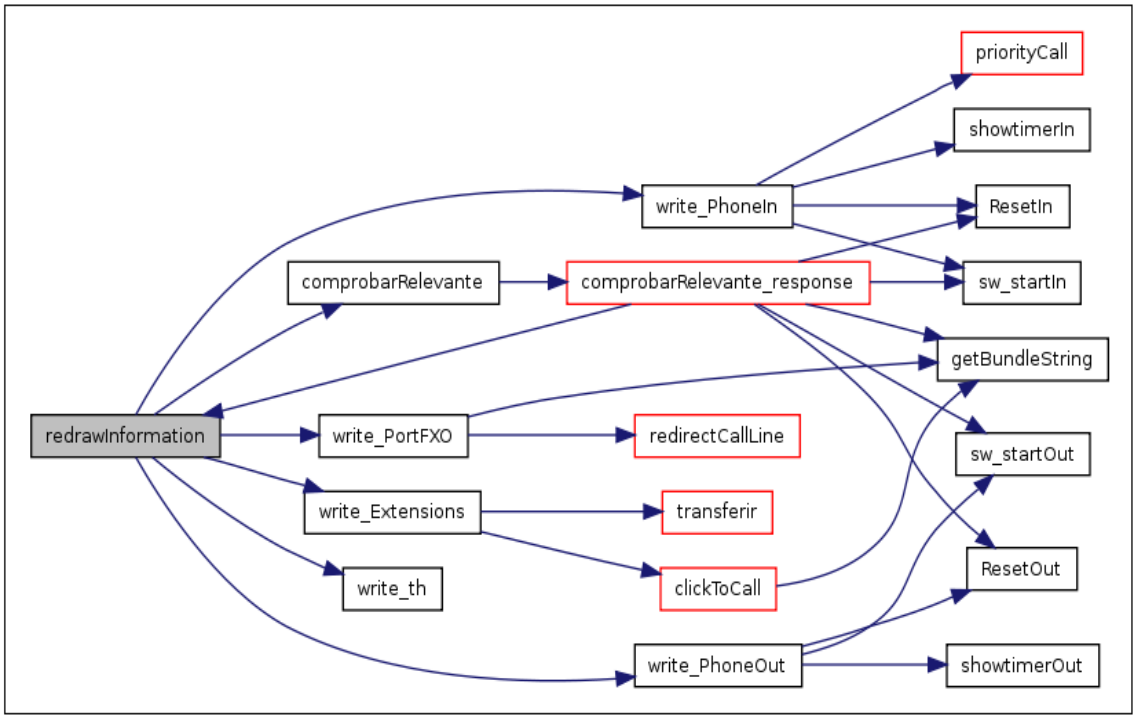

Figura 3.25: Grafo de llamada de la función *redrawInformation*.

Tras la realización de la actualización, los módulos vuelven a realizar la demanda y se quedan a la espera de respuesta. El ciclo descrito se sigue a lo largo de toda la jornada de trabajo del usuario operador encargado de la monitorización del sistema.

3.3.4.4. Módulo de envío de acciones.

La función del módulo de envío de acciones reside en permitir la interacción con la centralita *PBX*. Al contrario que el módulo del procesado de eventos, que analiza de forma pasiva la actividad de la centralita, este módulo permite alterar su funcionamiento enviando órdenes a *Asterisk*. En la Figura 3.26 se puede observar el diagrama de flujo correspondiente a la actividad del módulo de envío de acciones.

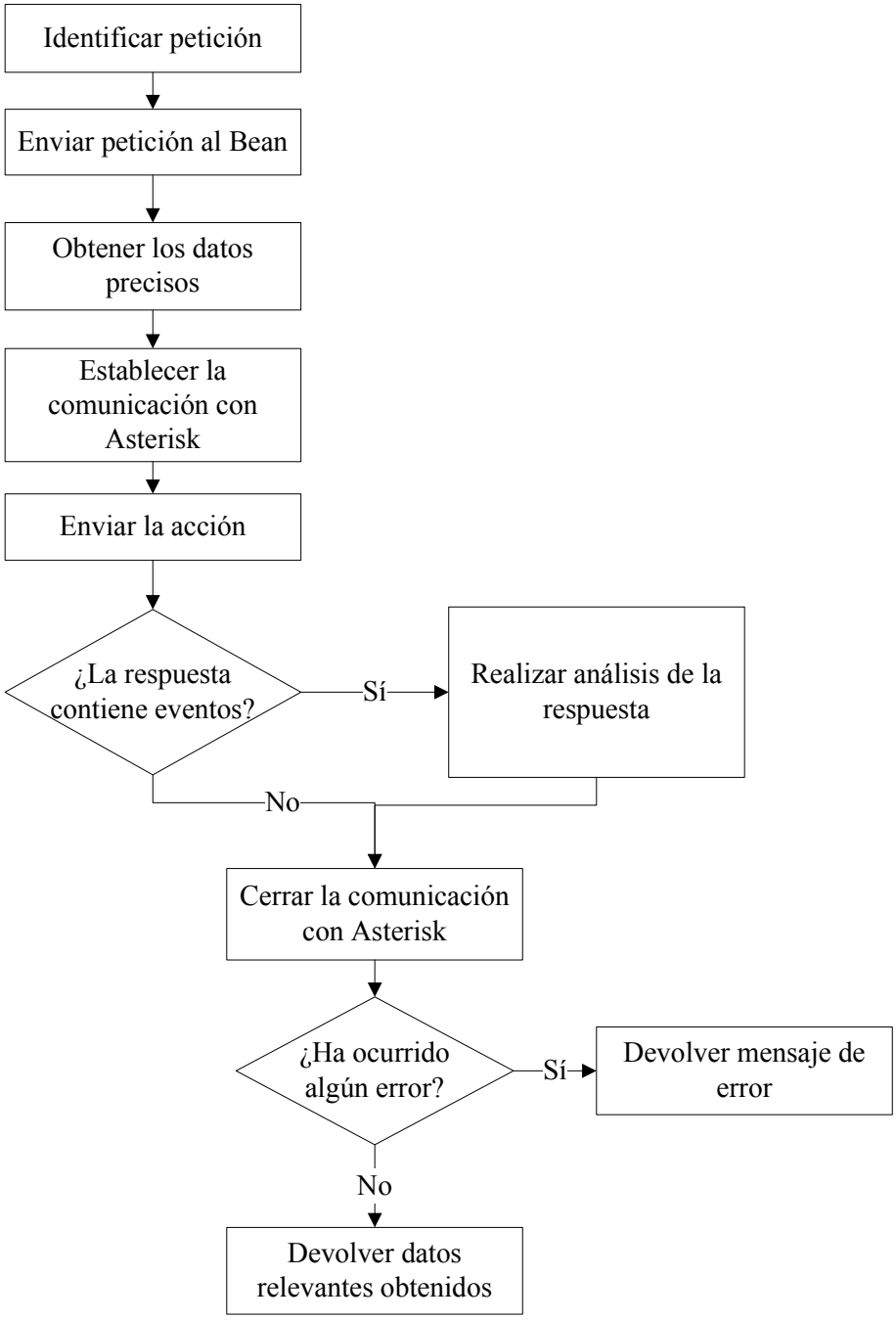

Figura 3.26: Diagrama de flujo del Módulo de Envío de acciones.

Como se puede observar, el presente módulo se encarga de estar a la escucha de las posibles peticiones que se generen en la página web debidos a la interacción con el usuario. Una de las funciones de los módulos de visualización es permanecer a la escucha de eventos causados por el usuario, que pueden ser arrastrar cierto elemento de la página o simplemente realizar un doble click sobre un componente de la página web. Cuando se produce alguno de esos eventos, el componente de visualización correspondiente solicita la intervención del módulo de envío de acciones porque la situación descrita equivale a la petición del usuario de mandar una orden a la centralita.

Siguiendo el diagrama de flujo presentado, se puede observar que lo primero es identificar el tipo de petición que trata de realizar el usuario. Una vez llevada a cabo la identificación, se envía la petición a *AsteriskManagerBean* que es la clase java encargada de la implementación de todas las interacciones directas con el manager de *Asterisk*. A continuación, dependiendo de la petición, se hace necesaria la obtención de cierta información adicional sobre determinados componentes del sistema que normalmente se realiza a través de la comunicación con el módulo externo de la base de datos.

En el momento en el que ya se disponen de todos los datos requeridos, la aplicación se encuentra en disposición de construir la acción correspondiente y enviársela al manager de *Asterisk*. Algunas de las acciones proporcionadas por la librería *Asteriskjava* que se emplean en el presente proyecto son las siguientes:

- *CommandAction.*
  La presente acción ofrece la posibilidad de enviar al manager de *Asterisk* cualquier comando de los disponibles para su uso desde la consola. Es una de las acciones que más se utilizó para la implementación del proyecto ya que su gran versatilidad es muy útil para la obtención de información muy precisa sobre el estado de diversos componentes del sistema.

- *OriginateAction.*
  Esta acción permite llevar a cabo diferentes aplicaciones y se emplea, por ejemplo, para originar una llamada mediante el uso de aplicación *Dial*.

- *ParkAction.*
  Esta acción permite realizar el aparcado de una llamada que el usuario considere en cierto instante menos prioritaria que otras. El interlocutor pasa a escuchar música en espera hasta que el usuario operador decida recuperar la llamada aparcada.

- *RedirectAction.*
  Esta acción permite la realización de una transferencia de una llamada.

- *StatusAction.*
  Esta acción se usa para solicitar información sobre el estado de los canales activos del sistema. Se emplea junto con *EventGeneratingAction* para provocar que el manager de *Asterisk* conteste a la petición enviando una colección de eventos tipo *StatusEvent*, es decir, envía un evento por cada canal activo que exista en ese instante en el sistema.

Como se ha comentado más arriba, existen acciones a las que el manager contesta con una colección de eventos, que no forman parte de los generados como fruto de la actividad normal y por tanto deben ser analizados de forma separada. Entonces, si se trata de este caso, el módulo de envío de acciones se encarga de analizar la respuesta recibida, en los demás casos simplemente la respuesta es un mensaje de éxito o error. A continuación, se cierra la conexión con el manager de *Asterisk* y en caso de que la orden se haya procesado correctamente, se envía la información obtenida como respuesta a la petición de uno de los submódulos de visualización. En caso de que haya ocurrido algún fallo, se envía un mensaje de error, que el módulo correspondiente deberá presentar al usuario para advertirle de lo ocurrido.

3.4. Configuración.

Como ya se ha comentado en ocasiones anteriores, el software desarrollado no es una aplicación independiente sino que forma parte de un proyecto mayor que consiste en la implementación de toda la inteligencia de red necesaria para una *PBX*. Lo cual significa que el código está integrado dentro de la aplicación de la *IRI* y la instalación se hace del producto total.

A lo largo de este apartado se explicarán las configuraciones que es necesario realizar para el uso de la aplicación, sin embargo, se obviarán otras muchas configuraciones imprescindibles para el funcionamiento de la totalidad de la centralita pero que no afectan directamente a la herramienta presentada.

3.4.1. Configuración de *Asterisk*.

Como se comentó en el estado del arte, *Asterisk* ofrece numerosas prestaciones y funcionalidades pero en este apartado se explicará la configuración de la centralita necesaria para conseguir el correcto funcionamiento de la herramienta, sin abarcar los puntos referentes a las demás posibles prestaciones.

También se ha mencionado en la descripción realizada de las características de *Asterisk*, que la configuración del sistema se realiza por ficheros, es decir, el comportamiento de la centralita deseado se obtiene mediante la introducción de ciertos parámetros de los ficheros correspondientes y realizando el análisis de éstos cuando se carga, *Asterisk* obtiene la información necesaria para la definición de su comportamiento. Sin embargo, existe el sistema denominado *realtime*, que permite el almacenamiento de la configuración de *Asterisk* en una base de datos en lugar de en los ficheros. De esta forma, la configuración del sistema puede ser accedida en tiempo real en lugar de tener que hacerlo en el momento de cargar los módulos. Por lo tanto, este sistema permite realizar modificaciones de la configuración sin necesidad de reiniciar *Asterisk*.

La solución de inteligencia de red de la que forma parte la aplicación del terminal de operador, se basa precisamente en un *Asterisk realtime*, y además ofrece una interfaz web que permite realizar la configuración de la *PBX* de una forma mucho más sencilla e intuitiva.

Dicha interfaz permite crear usuarios y agruparlos según el comportamiento deseado. Ofrece la posibilidad de crear pasarelas de voz que pueden estar implementadas por un router, por una tarjeta analógica o una tarjeta *RDSI*. Es capaz de facilitar la configuración de líneas de entrada, redirecciones de llamadas, salas de conferencias, macros y grupos de captura.

En lo referente al presente proyecto, la configuración de *Asterisk* es necesaria pero no forma parte del trabajo como tal. Por ello en este apartado sólo se mencionan las partes de la configuración características del terminal de operador.

Lo más importante de esta parte de la configuración es indicarle al manager de *Asterisk* que debe enviar todos los eventos relevantes. Es decir, por defecto no envía todos los eventos relevantes para determinar en todo momento el estado de la centralita. Para conseguir dicho propósito es necesario asegurar que el parámetro denominado *callevents* se encuentre inicializado a *yes*. Normalmente dicho parámetro debería encontrarse en el archivo *sip.conf*, en caso de tratarse de una centralita con *Asterisk realtime*, debe almacenarse en la tabla que alberga las demás características *SIP*.

También es importante establecer el comportamiento de las llamadas aparcadas. El terminal de operador ofrece la posibilidad de elegir una de las llamadas entrantes para darle prioridad. Es decir, el operador pasa a atender la llamada escogida mientras que el resto de las llamadas entrantes son aparcadas en una extensión auxiliar. Cuando el operador decide recuperar alguna de las llamadas aparcadas, *Asterisk* se encarga de restaurarla dirigiéndose a la extensión en la que se encuentra aparcada. Para permitir el comportamiento descrito es necesario crear un patrón determinado y asignarle las acciones adecuadas.

En este caso de creó un patrón denominado *park*. Por defecto, *Asterisk* reserva las extensiones del rango 701-720 para proporcionar la funcionalidad descrita, por lo tanto, el patrón creado fue 7XX.

A continuación, se definieron dos aplicaciones que deben ejecutarse con su correspondiente prioridad. Con la prioridad 1 debe ejecutarse la aplicación *Goto* con los siguientes argumentos:

$$parkedcalls \mid \${EXTEN : 0} \mid 1$$

Con la prioridad 2, se debe ejecutar la aplicación *Hangup*.

Es importante asignar el contexto creado a todos los grupos de usuarios cuyos miembros se deseen monitorizar con la herramienta del terminal de operador.

Otro punto a tener en cuenta es que sólo se pueden monitorizar usuarios que disponen de alguna extensión asignada, es por tanto imprescindible, adjudicarle una extensión tras la creación de los usuarios.

3.4.2. Configuración de las tarjetas.

Inicialmente, el paquete que formaba parte de *Asterisk* cuya función residía en el manejo de los dispositivos hardware externos, se llamaba *Zaptel*. Sin embargo, debido a problemas derivados del nombre, ya que existe una empresa que lo comparte, los desarrolladores de *Digium* lanzaron un nuevo paquete denominado *DAHDI* (*Digium Asterisk Hardware Device Interface*). A parte del cambio de nombre también introduce diversas mejoras de funcionalidad. Para la realización del presente proyecto se empleó esta nueva versión del paquete mencionado.

Para llevar a cabo el presente proyecto se emplearon diversas tarjetas y se estudió la respuesta de la herramienta ante el uso de todas ellas. Concretamente las tarjetas empleadas, como ya se comentó en el apartado de herramientas empleadas, fueron las siguientes:

- Xorcom Astribank 2 *BRI* (XR0013)

  Se trata de una tarjeta que proporciona dos accesos básicos *RDSI*, es decir, accesos simultáneo a dos canales de 64*Kbps* para voz o datos y un canal de 16*Kbps* para la señalización.

- Xorcom Astribank 8 FXO (XR0019)

  Este componente ofrece el hardware necesario para proporcionar la conexión a la centralita una colección de 8 líneas analógicas.

- Xorcom Astribank 8 FXS (XR0001)

  Por último, se empleó el dispositivo capaz de permitir la conexión de 8 teléfonos analógicos o fax.

Hay que resaltar que son unas tarjetas especialmente diseñadas para funcionar con *Asterisk*, sin embargo, no son obligatorias ni éstas, ni las fabricadas por *Digium*, sino que existen numerosas tarjetas compatibles.

Para realizar la configuración de los dispositivos expuestos, hay que realizar la modificación pertinente de dos archivos, que sustituyen a *zapata.conf* empleado en las versiones antiguas de la configuración de las tarjetas *Zaptel*. Los dos archivos implicados se localizan en el directorio */ect/asterisk*. El primero de esos archivos característicos del controlador *DAHDI* es

el denominado *chan_dahdi.conf* es el encargado de albergar las propiedades globales de un grupo de canales. Un ejemplo de la configuración de un grupo de canales empleado para la realización del proyecto se presenta a continuación.

[channels]

language=es
context=IncomingCalls
switchtype=national
rxwink=300
usecallerid=yes
hidecallerid=no
callwaiting=yes
usecallingprogres=yes
callwaitingcallerid=yes
threewaycalling=yes
transfer=yes
cancallforward=yes
callreturn=yes
echocancel=yes
echocancelwhenbridged=no
rxgain=0.0
txgain=0.0
group=1
callgroup=1
pickupgroup=1
inmediate=no
answeronpolarityswitch=yes
hanguponpolarityswitch=yes

La configuración mostrada corresponde a un grupo que pertenece al contexto de *IncomingCalls* y en el cual se definen características que permitirán la actividad deseada., como por ejemplo la cancelación del eco, permitir la transferencia, indicar el identificador del llamante, etc.

El segundo archivo que interviene en la definición del comportamiento de las tarjetas es el denominado *dahdi-chanels.conf* que es el que debe albergar las propiedades específicas de las que pueden gozar los canales que forman el grupo definido. A continuación se adjunta un extracto del archivo mencionado.

signalling=fxs_ks
callerid=asreceived
group=1
context=IncomingCalls
channel => 1

signalling=fxs_ks
callerid=asreceived
group=1
context=IncomingCalls
channel => 2

La diferencia entre un canal *FXO* y un canal *FXS* reside en qué parte de la conexión proporciona el tono de marcar. Esa característica explica el hecho de que para realizar la

configuración de estos dos tipos de canales hay que tener en cuenta que cambia el parámetro de *signalling*. Es decir, la configuración presentada arriba corresponde a dos canales *FXO* y para conseguir el comportamiento de un canal como *FSX* habría que indicar en la configuración el parámetro *signalling=fxo_ks*.

En el mismo archivo se realiza también la definición del comportamiento de los canales *RDSI* del sistema. Existen numerosos parámetros configurables en cada caso, sin embargo, para el comportamiento por defecto, basta con indicar que la señalización utilizada corresponde a un canal *RDSI*, diferenciándose en esta ocasión los casos de enlaces básicos y primarios.

3.4.3. Configuración de los teléfonos *IP*.

La forma más cómoda y fácil de realizar la configuración de un teléfono *IP* es a través del interfaz web que proporcionan los fabricantes. El primer paso a realizar es asignarle una dirección *IP* al teléfono de interés y a partir de ese punto será accesible vía web simplemente mediante la introducción de dicha dirección en el navegador que se desee. Hay algunos modelos que permiten dicho acceso mediante algún puerto que debe estar reflejado en las instrucciones de uso, en otros, con la dirección *IP* es suficiente. Para permitir el acceso se pedirá la identificación del usuario que debe ser proporcionada por el fabricante. Como ya se ha indicado con anterioridad, a lo largo de la realización del proyecto se han empleado diversos modelos de teléfono *IP*, para realizar una breve explicación del proceso una configuración básica, se ha escogido el modelo DI-Voz que es uno de los que ofrece un interfaz más sencillo.

Una vez se ha accedido al interfaz, se debe realizar la configuración de la red, para ello se dispondrá de una vista similar a la que aparece en la Figura 3.27.

Figura 3.27: Interfaz de inicio de configuración de un teléfono IP.

La presente pantalla permite al usuario modificar la dirección *IP* asignada o elegir la opción de asignación de ésta por *DHCP* (*Dynamic Host Configuration Protocol*). También hay que introducir la dirección *IP* de la pasarela empleada y la máscara de red, a la vez que el *DNS* (*Domain Name System*) empleado. Una vez realizada la configuración de red, es el turno de la configuración *SIP* del teléfono, en esta ocasión, la vista proporcionada demandará información sobre la localización de la centralita y tendrá un aspecto como el que se representa en la Figura 3.28.

Figura 3.28: Interfaz de configuración de características SIP.

Los datos más importantes que se deben introducir son la dirección del servidor de registro, que es la dirección donde se alberga la centralita, y que en este caso es 192.168.18.10. El puerto que usa el protocolo *SIP* por defecto es el 5060 y es el que se usa en esta ocasión. También es relevante el tiempo de registro, que es el intervalo que se desea asignar para que el teléfono renueve su registro en la centralita, su valor por defecto es de 3600 segundos aunque en esta ocasión, como se puede observar, se ha empleado un valor menor, concretamente de 1000 segundos.

El siguiente paso es la configuración de la cuenta de usuario, en este caso será necesario introducir los datos del usuario *SIP* que previamente ha sido registrado en la centralita. Para ello tan sólo hay que reproducir los datos identificativos del usuario almacenados en la configuración de la *PBX*.

Como se puede observar en la pantalla presentada en la Figura 3.29 se ha realizado la configuración de la cuenta de un usuario *SIP* registrado con el nombre de *usuario* en la centralita.

Figura 3.29: Interfaz de configuración de cuenta de usuario.

Una vez introducidos los datos necesarios, al guardar los cambios, se manda la petición de registro a *Asterisk* y si todo transcurre correctamente, en el campo "*Register Status*" aparecerá *Register*, como se puede observar en la imagen presentada, indicando que el terminal se encuentra listo para su uso.

El interfaz ofrece otros muchos aspectos que se pueden configurar pero para conseguir la operatividad del terminal los pasos descritos son suficientes, por lo tanto, para finalizar la configuración tan sólo hay que reiniciar el aparato para asegurar el almacenamiento de la configuración. Dicho proceso se puede realizar también vía web o manualmente.

3.4.4. Configuración del Terminal de Operador.

Para comenzar con el trabajo de monitorización del sistema, aunque se dispone de una configuración por defecto, sería aconsejable realizar una configuración del terminal de operador para adaptarlo lo máximo posible a las necesidades del usuario. El proceso que se debe seguir para llevar a cabo dicha configuración y las facilidades que ofrece la aplicación se describen en el Manual de usuario presentado en el capítulo 4 de la presente memoria.

3.5. Pruebas realizadas.

El propósito de este apartado es realizar una breve descripción de las pruebas que se ejecutaron a lo largo del desarrollo del presente proyecto.

En la etapa inicial del diseño, se ejecutaron pruebas para familiarizarse con el funcionamiento de una centralita *Asterisk*. En la etapa del desarrollo se fueron realizando pruebas para comprobar el funcionamiento deseado y por último, una vez cerrada una versión del producto, se desarrolló una batería de pruebas para asegurar el cumplimiento de la especificación y la robustez del producto.

Antes de proceder con el diseño definitivo de la herramienta, fue necesario el análisis exhaustivo de los eventos que genera una centralita *Asterisk* durante su actuación normal. Para ello, el primer paso fue la construcción del servicio encargado de la escucha de los eventos, descrito en anteriores apartados, con el fin de facilitar el estudio de los eventos necesario. Una vez desarrollado el servicio capaz de permanecer a la escucha de los eventos y de realizar la implementación del código que permitía la obtención de los datos más relevantes de cada evento recibido, el siguiente paso consistió en realizar diversas pruebas para observar de qué información se disponía exactamente como resultado de las diferentes situaciones.

Para ello se realizaron principalmente pruebas sencillas, como el establecimiento de una llamada, su interrupción o su transferencia a otro usuario. Gracias a la observación de la actividad de la centralita a través de los eventos generados, se pudo realizar el diseño de la parte de la aplicación que se encarga del análisis de los eventos y de la traducción de los datos obtenidos en información sobre el estado de la centralita mostrado al usuario.

Cada vez que se finalizaba la implementación de una de las funcionalidades, se realizaban pruebas pertinentes para la comprobación de su correcto funcionamiento.

Una vez se completó la elaboración del código, se definió un banco de pruebas para testear la funcionalidad de la aplicación desarrollada. Para elaborar dicho banco de pruebas, se recurrió a la definición de objetivos de la construcción de la herramienta y se consideró necesaria la realización de las pruebas que se detallan a continuación.

3.5.1. Llamadas del sistema.

Para comprobar el correcto funcionamiento de la aplicación con respecto a las llamadas del sistema se realizaron pruebas de todo tipo de llamadas. A continuación se representan algunas figuras como justificante de haber obtenido el resultado esperado.

- Llamada entrante al sistema atendida.

Figura 3.30: Llamada entrante atendida.

En la Figura 3.30 aparece la representación de la herramienta de una llamada entrante al sistema. Se puede observar como se refleja la información sobre el número de teléfono que originó la llamada, el usuario del sistema destinatario de la misma y el tiempo transcurrido desde la conexión de los dos canales que intervienen.

- Llamada entrante al sistema perdida.

En la Figura 3.31 se representa el inicio de una llamada entrantes al sistema. La configuración de la centralita establece que el terminal telefónico del usuario destinatario de la llamada suene un tiempo determinado. En el caso representado, tras la expiración de dicho tiempo el usuario destinatario vuelve a encontrarse libre como se observa en la Figura 3.32.

Figura 3.31: Inicio de una llamada entrante al sistema.

Figura 3.32: Llamada entrante al sistema perdida.

- Llamada interna atendida.

En la Figura 3.33 se presenta la forma de reflejar una llamada interna de la aplicación realizada. Como una llamada interna no goza de la misma relevancia que las llamadas del sistema, la información relevante tan sólo se ve reflejada en la pantalla dedicada a las extensiones del sistema. Así se puede observar que los usuarios involucrados se representan como ocupados y se muestra el interlocutor correspondiente en cada caso.

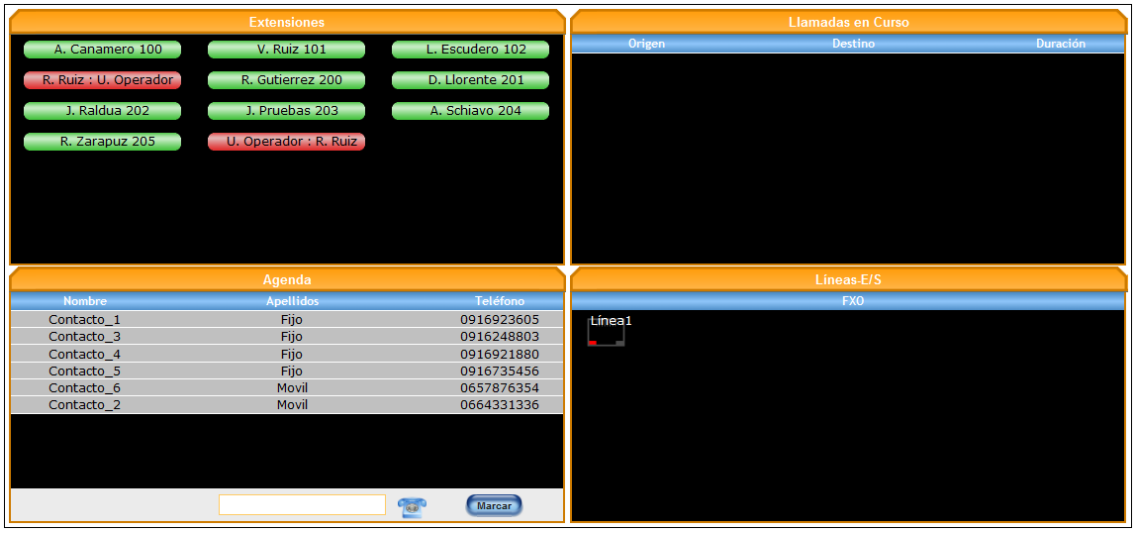

Figura 3.33: Ilustración de una conversación entre dos usuarios del sistema.

En el presente apartado se describen pruebas básicas de la respuesta del sistema, por lo tanto no se considera necesaria la justificación gráfica de todas ellas. En otras pruebas, se realizó la comprobación del comportamiento de la interfaz ante los siguientes casos. En todos ellos se obtuvo la representación de la situación con un retardo mínimo con respecto a la realidad.

- Llamada interna perdida.
- Llamada saliente del sistema atendida.
- Llamada saliente del sistema perdida.

3.5.2. Establecimiento de llamadas mediante el terminal.

La herramienta del terminal de operador permite el establecimiento de llamadas a través de la interacción con la interfaz gráfica. Las explicaciones detalladas del método de generación de dichas llamadas se presentan el capítulo del Manual de usuario.

En este apartado se presenta el resultado obtenido ante el establecimiento de las diferentes llamadas.

- Establecimiento de una llamada con un usuario del sistema.

Para generar la llamada representada en la Figura 3.34, el usuario operador realizó un doble click sobre la extensión del usuario con el que deseaba comunicarse.

Figura 3.34: Establecimiento de una llamada con otro usuario del sistema.

- Establecimiento de una llamada con un número de la agenda.

En el siguiente caso, el usuario operador pretendía establecer una conversación con un contacto registrado en su agenda. En esta ocasión también basta con un doble click sobre la entrada de la agenda correspondiente. En la Figura 3.35 se representa la situación descrita y se puede observar que se genera la llamada deseada con el contacto registrado como "Contacto_2".

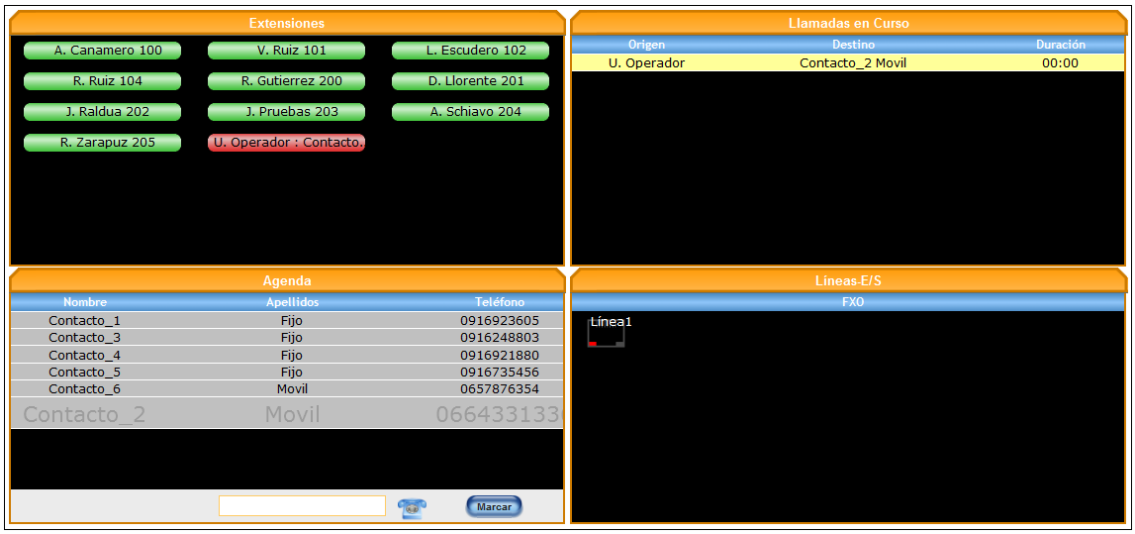

Figura 3.35: Establecimiento de una llamada con un contacto registrado.

- Establecimiento de una llamada con un número no registrado en la agenda.

También es posible establecer llamadas con números de teléfono no registrados en la agenda del usuario, para ello hay que introducir el número en cuestión en el espacio reservado tal y como aparece en la Figura 3.36. A continuación basta con presionar el botón de "Marcar" y como resultado se observa la generación de una llamada saliente desde el terminal del usuario operador, con el número deseado.

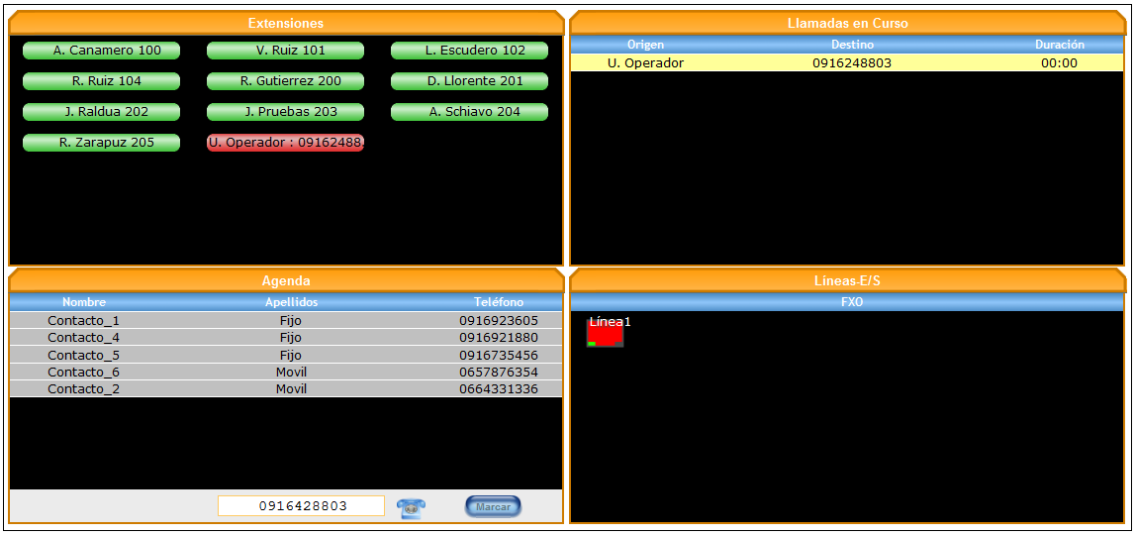

Figura 3.36: Establecimiento de una llamada con un número no registrado en la agenda.

- Establecimiento de una llamada empleando una línea de salida en concreto.

La herramienta permite también el encaminamiento de las llamadas por las líneas que desee el usuario operador. Dicha operación también es posible tanto con los números registrados en la agenda del usuario, como con los que no lo están.

En la Figura 3.37 se representa el intento del establecimiento de una llamada con el número registrado como "Contacto_3".

Figura 3.37: Intento de encaminamiento de una llamada por la línea seleccionada.

En la Figura 3.38 se puede observar el resultado de la operación. Es importante resaltar que el momento de la realización de las pruebas que sirvieron para la elaboración de la presente memoria tan sólo se disponía de una línea analógica, la cual aparece representada en las figuras obtenidas. Pero dicho hecho no invalida la prueba presentada ya que aparte de la línea analógica, la centralita estaba configurada para trabajar con un router que se encargaba de gestionar las llamadas a móviles, mientras que la línea analógica se reservaba para las llamadas a números fijos. Dicho hecho se confirma con las pruebas gráficas presentadas ya que se puede observar en todas las figuras que en caso de tratarse de llamadas a fijos, la línea *FXO* aparece como ocupada, mientras que en caso de llamadas a móviles aparece como libre. Se puede comprobar lo expuesto por ejemplo en la Figura 3.35 y en la Figura 3.36.

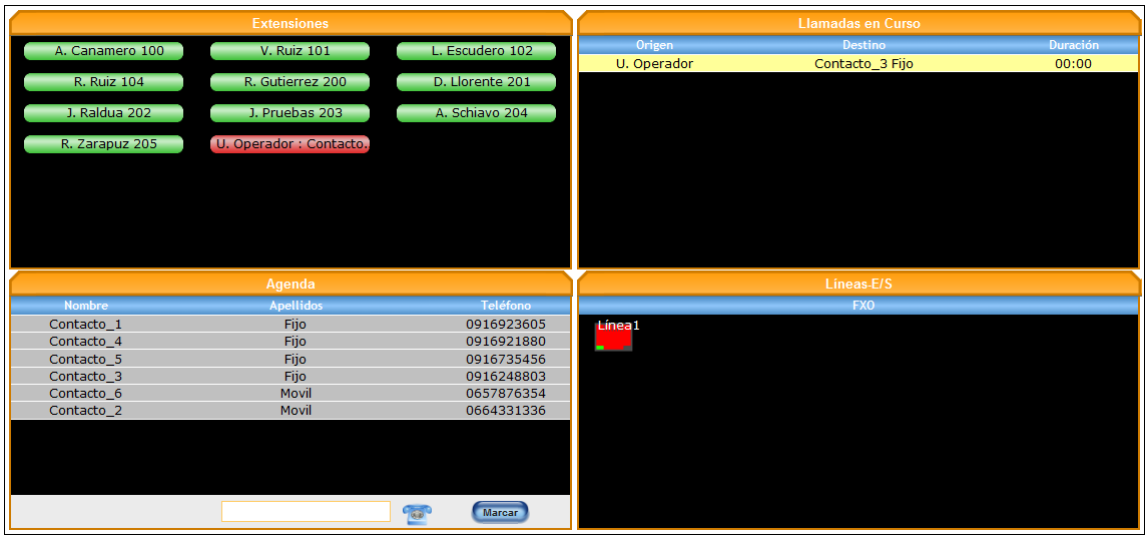

Figura 3.38: Establecimiento de una llamada por una línea en concreto.

3.5.3. Transferencia de llamadas del sistema.

En este apartado se pretende dejar constancia de la respuesta fehaciente de la herramienta ante las distintas transferencias que puedan llegar a realizar los usuarios del sistema con sus terminales telefónicos.

- Transferencia de una llamada entrante.

En las siguientes figuras se representa paso por paso la transferencia con consulta de una llamada entrante a otro usuario del sistema.

Figura 3.39: Inicio de una llamada entrante al sistema.

Figura 3.40: El usuario destinatario responde a la llamada entrante al sistema.

Figura 3.41: El usuario destinatario pone a la llamada entrante en espera.

Figura 3.42: El usuario destinatario intenta contactar con otro usuario del sistema.

Figura 3.43: El usuario destinatario ha transferido la llamada entrante a otro usuario del sistema.

Se comprobó también el comportamiento del terminal ante las siguientes situaciones:

- Transferencia de una llamada saliente.
- Transferencia de una llamada interna.

Se realizaron distintas pruebas de transferencias tanto hacia otros usuarios del sistema como hacia números de teléfonos ajenos a la centralita, obteniendo el resultado esperado en todos los ensayos realizados.

3.5.4. Transferencia de llamadas mediante el terminal.

La herramienta del terminal de operador permite realizar transferencias sin consulta previa, mediante la interacción con la interfaz. Dichas transferencias pueden ser tanto hacia otros usuarios del sistema como hacia los números de teléfono registrados en la agenda.

En caso de una llamada entrante al sistema, el usuario debe seleccionar con el ratón el origen de la llamada y arrastrarlo hasta el usuario o la entrada de la agenda hacia la cual desea transferir la llamada. En las figuras sucesivas se representa el proceso paso a paso.

- Transferencia ciega de una llamada entrante hacia otro usuario del sistema.

Figura 3.44: Inicio de una llamada entrante.

Figura 3.45: El usuario operador selecciona el origen de la llamada y lo arrastra hacia otro usuario del sistema

Figura 3.46: El usuario operador suelta el ratón y se realiza la transferencia de la llamada.

El siguiente caso documentado es el de una transferencia de una llamada saliente hacia un número de la agenda. En este caso, el usuario debe seleccionar el destino de la llamada y arrastrarlo hacia la entrada de la agenda que le interese. En las figuras presentadas a continuación se puede visualizar todo el proceso.

- Transferencia ciega de una llamada saliente a un número de la agenda.

Figura 3.47: Inicio de una llamada saliente del sistema.

Figura 3.48: El usuario arrastra el número destino hacia una entrada de la agenda.

Figura 3.49: Como resultado de la transferencia se establece una nueva llamada entre dos usuarios que no pertenecen al sistema.

También se comprobó el comportamiento deseado de la herramienta ante los siguientes casos:

- Transferencia ciega de una llamada entrante a un número de la agenda.
- Transferencia ciega de una llamada saliente hacia otro usuario del sistema.

3.5.5. Priorización de una llamada entrante.

La herramienta le permite al usuario operador atender la llamada que éste considere oportuno, en todo momento, mediante la interacción con el interfaz. Para ello, el usuario operador debe posicionarse sobre la llamada que le interese y realizar un doble click con el botón izquierdo del ratón.

En la Figura 3.50 se representa la situación en la que se han generado dos llamadas entrantes al sistema y el usuario debe elegir cuál de ellas considera más importante. En la Figura 3.51 se aprecia que el usuario ha decido atender primero la segunda llamada entrante y para ello la selecciona con el ratón. Como consecuencia de dicha acción, el operador pasa a atender la segunda llamada mientras que la primera permanece aparcada, dicha situación se puede observar en la Figura 3.52.

- Selección de una llamada concreta para ser atendida de entre varias llamadas entrantes al sistema.

Figura 3.50: Generación de dos llamadas entrantes al sistema.

Figura 3.51: El usuario operador decide atender la segunda llamada generada.

Figura 3.52: El usuario se encuentra ocupado con la conversación priorizada.

- La recuperación de una llamada enviada a un segundo plano.

Una vez finalizada la conversación establecida, el usuario operador puede recuperar la llamada aparcada con doble click sobre la llamada en cuestión.

Figura 3.53: El usuario operador realiza un doble click sobre la llamada aparcada.

Figura 3.54: El usuario operador recupera la llamada anteriormente aparcada.

En la Figura 3.54 se visualiza la situación descrita, en la cual, el usuario operador ha recuperado la llamada que se encontraba aparcada. Se puede observar como el campo de Duración correspondiente pasa de mostrar que la llamada está aparcada a presentar el contador del tiempo transcurrido desde el inicio de la conversación.

3.5.6. Pruebas de monitorización.

En este apartado se han incluido pruebas que se refieren a la observación de la respuesta de la herramienta ante distintas situaciones que se pueden llegar a dar a lo largo del funcionamiento normal de la centralita.

- Retención de una llamada y su posterior recuperación.

En la Figura 3.62 se muestra la respuesta de la herramienta ante la retención de una llamada por el usuario. La llamada afectada pasa a mostrar que se encuentra retenida empleando para ello el campo reservado a la duración. Y el usuario pasa a adquirir el estado denominado como "Retenidas" que refleja el hecho de que todas las llamadas que le incumben se encuentran retenidas.

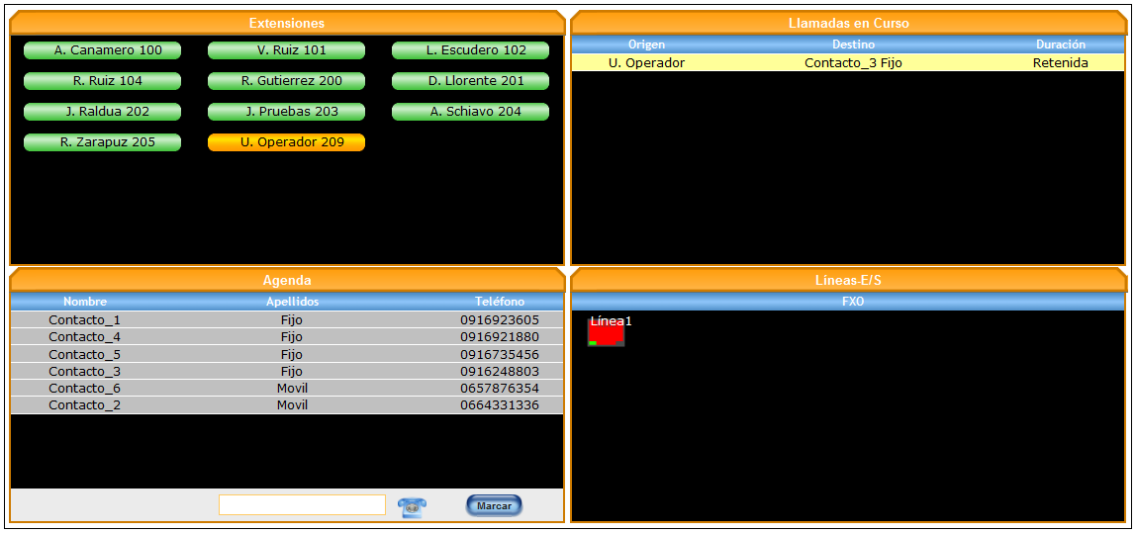

Figura 3.55: El usuario retiene una llamada saliente.

En la Figura 3.56 se observa la reacción del sistema en el momento en el que el usuario decide recuperar la llamada retenida. Es decir, en la pantalla de las extensiones se observa que el usuario vuelve a encontrarse ocupado en la conversación con el destinatario de la llamada que antes se encontraba retenida, mientras que el campo de la duración vuelve a mostrar el contador actualizado.

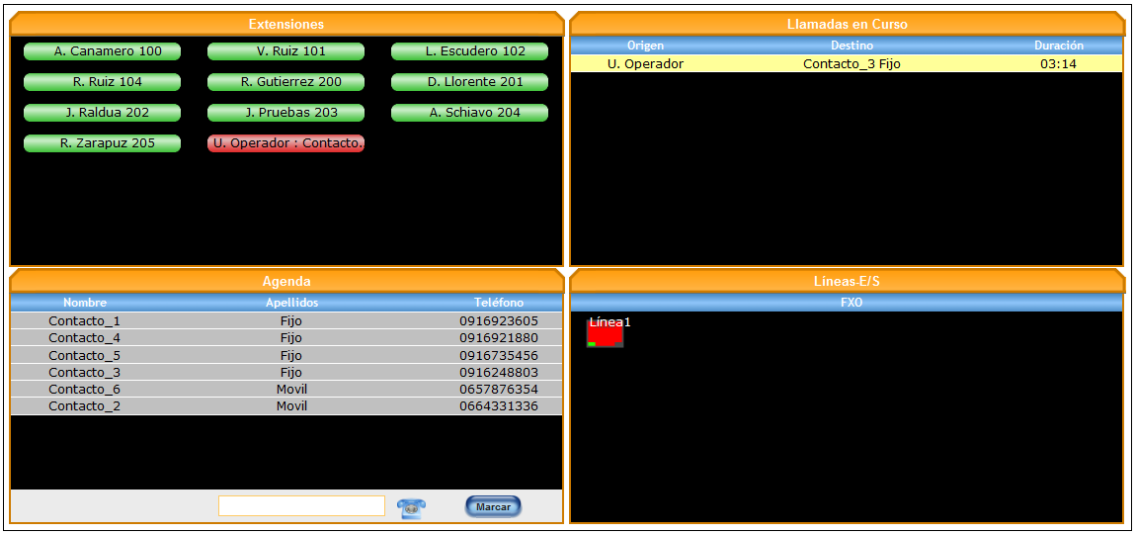

Figura 3.56: El usuario recupera la llamada saliente anteriormente retenida.

- Captura de una llamada desde otra extensión.

Una de las funcionalidades que ofrece una centralita *Asterisk* consiste en permitir la captura de las llamadas por usuarios que no son los destinatarios directos de éstas. Es decir, en caso de que se genere una llamada hacia una extensión en concreto que no está siendo atendida, otro usuario dispone de la posibilidad de interceptar dicha llamada. En la Figura 3.57 se observa como se ha generado una llamada entrante con destinatario U. Operador. Como dicho usuario no atiende la llamada, el usuario R. Gutiérrez intercepta la llamada presionando un conjunto determinado de teclas de su terminal telefónico y pasa a tender la llamada como se puede ver en la Figura 3.58.

Figura 3.57: Una llamada entrante para el usuario U. Operador.

Figura 3.58: La llamada entrante es recogida por el usuario R. Gutiérrez.

- Búsqueda de contactos en la agenda.

Para justificar el correcto funcionamiento de la funcionalidad de la búsqueda de contactos en la agenda del usuario primero se presenta la vista general de la agenda en la Figura 3.59.

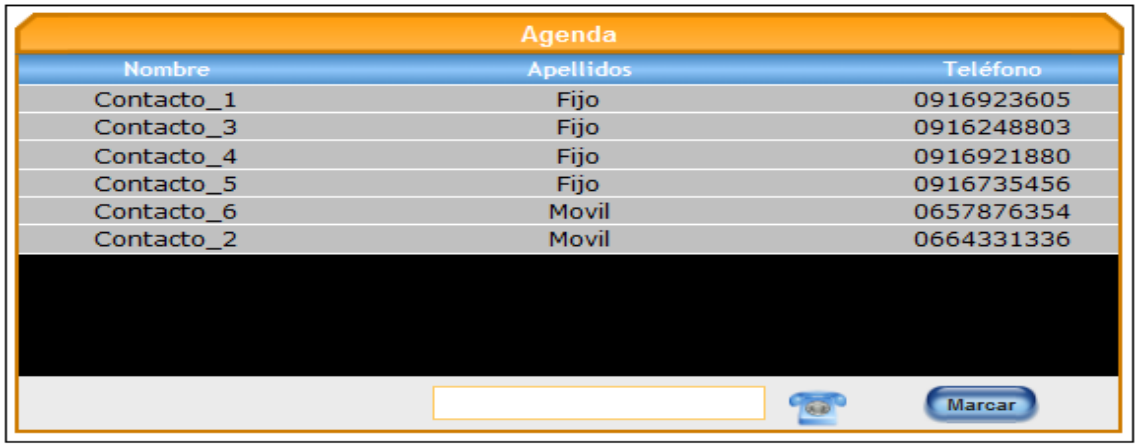

Figura 3.59: Vista general de la agenda del usuario.

A continuación se muestra el resultado de la consulta en la agenda de las entradas que coincidan con el patrón "fij" introducido en el espacio reservado a tal efecto en la parte inferior de la subpantalla como se puede observar en la Figura 3.60.

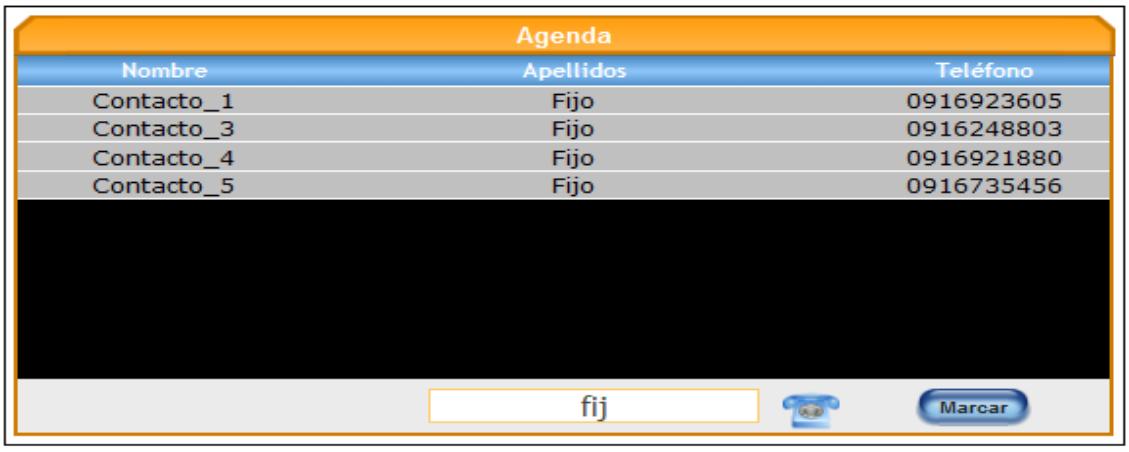

Figura 3.60: Resultado de la búsqueda del patrón "fij".

Como se ve, como resultado de la búsqueda se muestran todas las entradas de la agenda que cumplen con el patrón introducido, ocultando todos los contactos que no lo cumplen.

De la misma forma, en la Figura 3.61 se puede observar la respuesta de la herramienta ante la búsqueda de un patrón que no se encuentra en ninguna de las entradas de la agenda. Se realizaron otras diversas pruebas para confirmar el correcto funcionamiento de la presente parte de la herramienta.

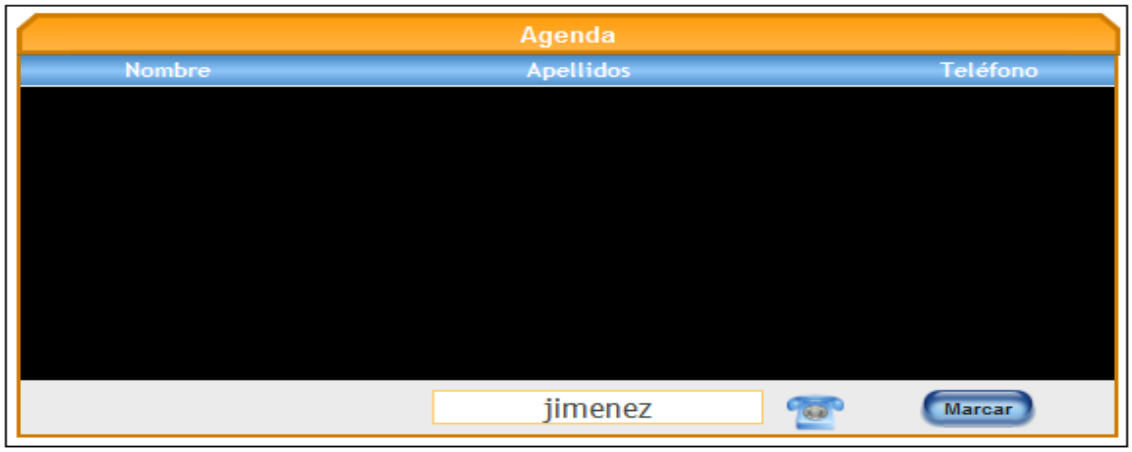

Figura 3.61: Resultado de la búsqueda del patrón "jimenez".

- Ingreso en una sala de conferencias.

La centralita *Asterisk* permite el establecimiento de llamadas entre varios interlocutores mediante las llamadas salas de conferencias. En la Figura 3.62 se representa la forma de la herramienta de registrar la participación de un usuario en la conversación de una sala de conferencia.

Figura 3.62: Ingreso del usuario a la sala de conferencias número 2.

Para confirmar la correcta respuesta del terminal de operador ante distintas situaciones del funcionamiento de la centralita se llevaron a cabo numerosas pruebas relacionadas con los siguientes casos:

- Monitorización del correcto funcionamiento de los contadores de la duración de la llamada.
- Monitorización del estado de los usuarios.
- Monitorización del estado de las líneas.
- Monitorización del estado de la centralita en el momento del ingreso en el sistema.
- Abandono de una sala de conferencias.

Tras la realización de dichas pruebas se puede afirmar que la herramienta es capaz de reflejar en todo momento de forma gráfica el estado actualizado de la centralita.

3.5.7. Introducción de credenciales.

Para justificar el correcto funcionamiento del módulo de autenticación se realizaron diversas pruebas introduciendo tanto datos correctos como incorrectos y observando la respuesta de la aplicación. En la Figura 3.63 se observa uno de los casos, en el que el usuario operador intenta acceder al sistema.

Figura 3.63: Acceso al sistema del usuario operador.

Como el usuario operador se encuentra registrado en el sistema y la clave introducida corresponde con la almacenada en la base de datos consultada, la herramienta permite la visualización del interfaz correspondiente al usuario que está realizando el acceso. Tal y como se puede observar en la Figura 3.64, se le muestra el menú correspondiente a su tipo de usuario y el terminal de operador correspondiente a la configuración que tenga almacenada.

En caso de introducir las credenciales de un usuario no registrado en el sistema o en caso de equivocarse al introducir la clave, la herramienta no permite el acceso al sistema mostrando un mensaje de error.

Figura 3.64: Vista del interfaz de inicio del usuario operador.

3.5.8. Configuración del interfaz.

Para constatar el correcto funcionamiento de la parte de la herramienta que se encarga de registrar las preferencias del usuario sobre el aspecto del terminal, se realizaron pruebas formalizando distintas configuraciones. A continuación se presenta la documentación de una de ellas.

En la Figura 3.65 se puede observar la distribución de la parrilla de extensiones que ha realizado el usuario. Mientras que en la Figura 3.66 se refleja que el usuario ha escogido la opción de que las extensiones se identifiquen mediante el nombre de los usuarios correspondientes y que las extensiones se presenten en la mitad superior de la pantalla completa del terminal de operador.

En la Figura 3.67 se visualiza el interfaz de operador correspondiente a la configuración realizada. Donde se puede observar la correcta respuesta de la herramienta ya que se presentan las pantallas seleccionadas, la distribución de las extensiones es la reflejada en la parrilla y las extensiones se identifican mediante la inicial del nombre y el primer apellido de los usuarios correspondientes.

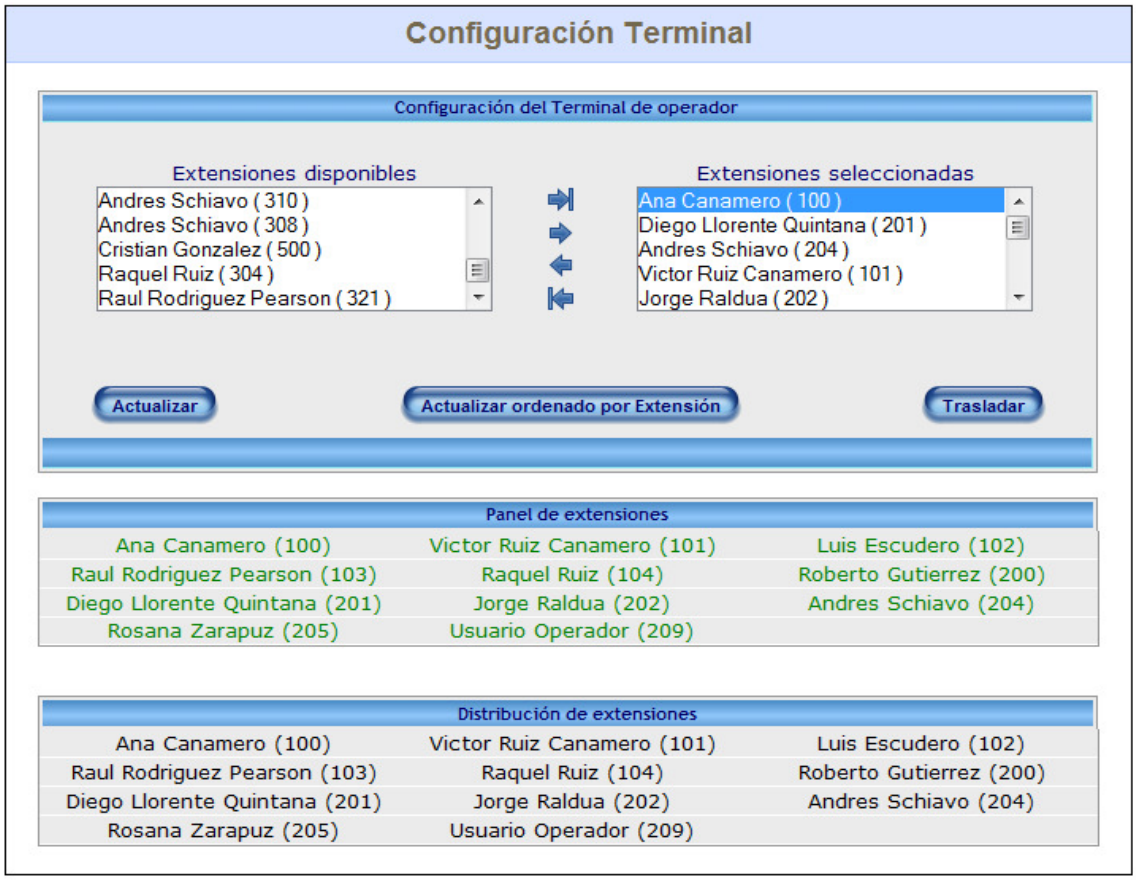

Figura 3.65: Parte de la configuración referente a la parrilla de las extensiones.

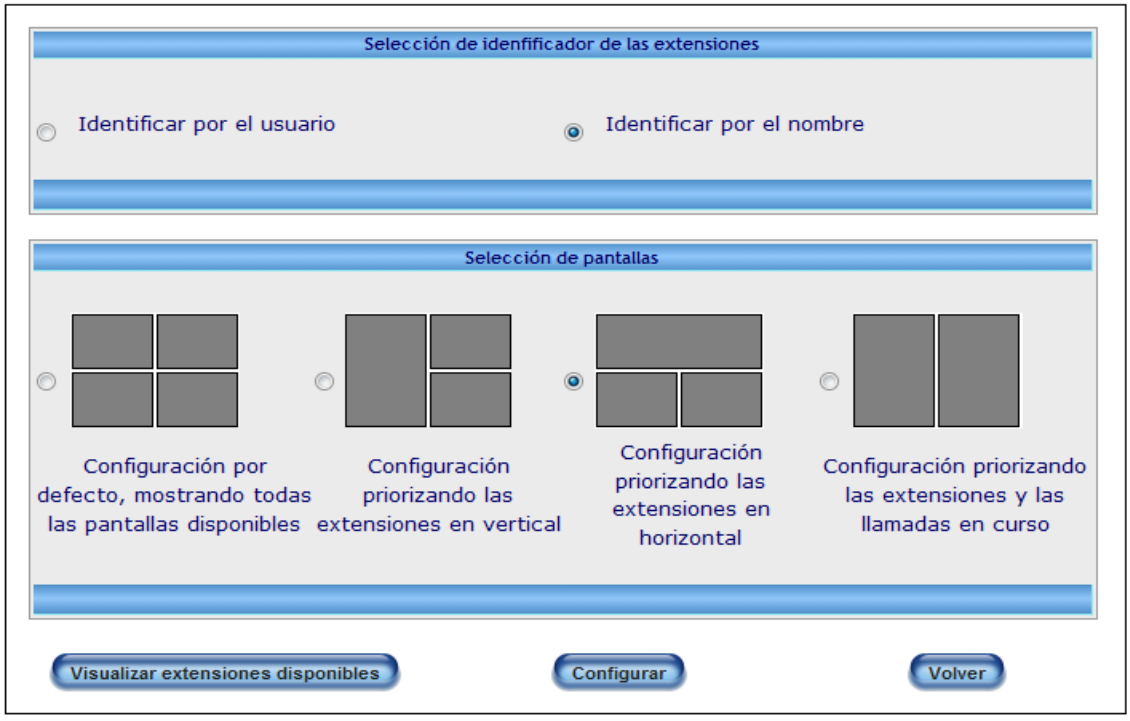

Figura 3.66: Parte de la configuración referente a la identificación de usuarios y selección de pantallas.

Figura 3.67: Terminal de operador correspondiente a la configuración realizada.

3.5.9. Pruebas de estrés.

Para la realización de las pruebas de estrés se empleó el programa *sipp* descrito en el apartado de las herramientas utilizadas.

Dicho programa permite generar llamadas con distintas características y con el destinatario deseado mediante una simple instrucción. Se realizaron numerosas pruebas para comprobar el comportamiento de la herramienta ante una gran afluencia de llamadas. A continuación se presentan algunas ilustraciones de los resultados obtenidos.

En la Figura 3.68 se puede observar la generación de numerosas llamadas con el mismo usuario como destinatario, respondiendo éste a la primera de ellas. En la Figura 3.69 describe la situación en la que el usuario se encuentra atendiendo la tercera llamada en generarse habiendo retenido el resto.

Figura 3.68: Generación de varias llamadas entrantes con el mismo usuario como destino.

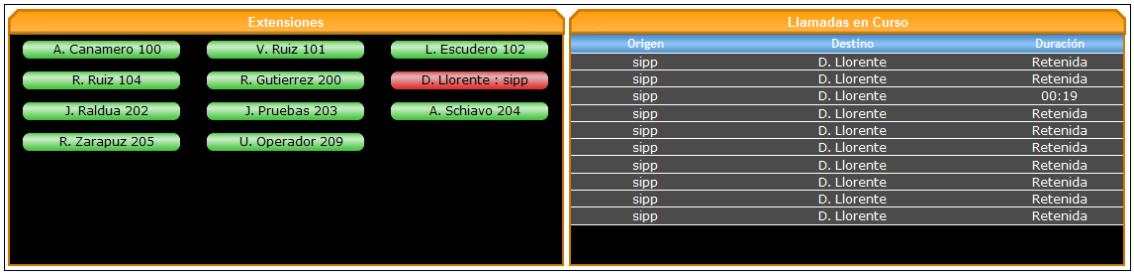

Figura 3.69: El usuario se encuentra ocupado con una de las llamadas reteniendo el resto.

En la Figura 3.70 se puede apreciar que el usuario ha finalizado la conversación que estaba manteniendo y en ese instante todas sus llamadas se encuentran retenidas, lo cual se refleja también en la pantalla de las extensiones.

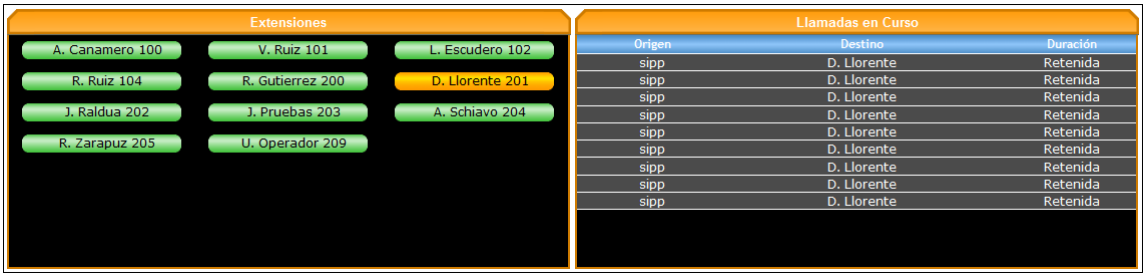

Figura 3.70: El usuario tiene todas sus llamadas retenidas.

En la situación representada en la Figura 3.71 se observa que el usuario se encuentra ocupado con una de las llamadas que ha recuperado y a la vez se está produciendo un intento de transferencia de otra de las llamadas.

Figura 3.71: Inicio de una transferencia de una de las llamadas retenidas.

En la Figura 3.72 se puede observar el resultado de la transferencia de la llamada, tras la cual, la llamada que se encontraba retenida para a ser atendida por otro usuario.

Figura 3.72: Resultado de la transferencia de la llamada.

En el apartado de las pruebas de estrés, se realizaron simulaciones de generación de hasta 50 llamadas simultáneas, obteniéndose una correcta respuesta de la aplicación ya que se lograba en todo momento el reflejo fiel de la situación acontecida.

CAPÍTULO 4:
MANUAL DE USUARIO

4. MANUAL DE USUARIO

4.1. Introducción.

El terminal de operador es la propuesta de *InTecDom* para la gestión de y manipulación de llamadas en tiempo real. Es una completa solución llave en mano, que permite gestionar las llamadas de una central telefónica de una forma sencilla, mediante un amigable entorno Web.

Este capítulo realiza una detallada descripción de la forma de uso de la herramienta del terminal de operador.

Es importante mencionar que el presente manual tan sólo abarca la identificación del usuario de tipo Operador y el uso de las funcionalidades de la herramienta de monitorización. Es decir, no entra en la descripción ni de la configuración, ni de uso de la centralita.

4.2. Introducción de las credenciales de usuario.

Cuando el usuario accede a la aplicación web mediante el navegador se le muestra la pantalla de acceso que solicita su identificación para permitir continuar con el proceso. Dicha pantalla de inicio se muestra en la Figura 4.1 y se puede observar que se proporcionan dos campos para la introducción de credenciales, que son *Nombre de Usuario* y *Contraseña*.
Las credenciales por defecto son:

- Nombre de Usuario: *operador*
- Contraseña: *operador*

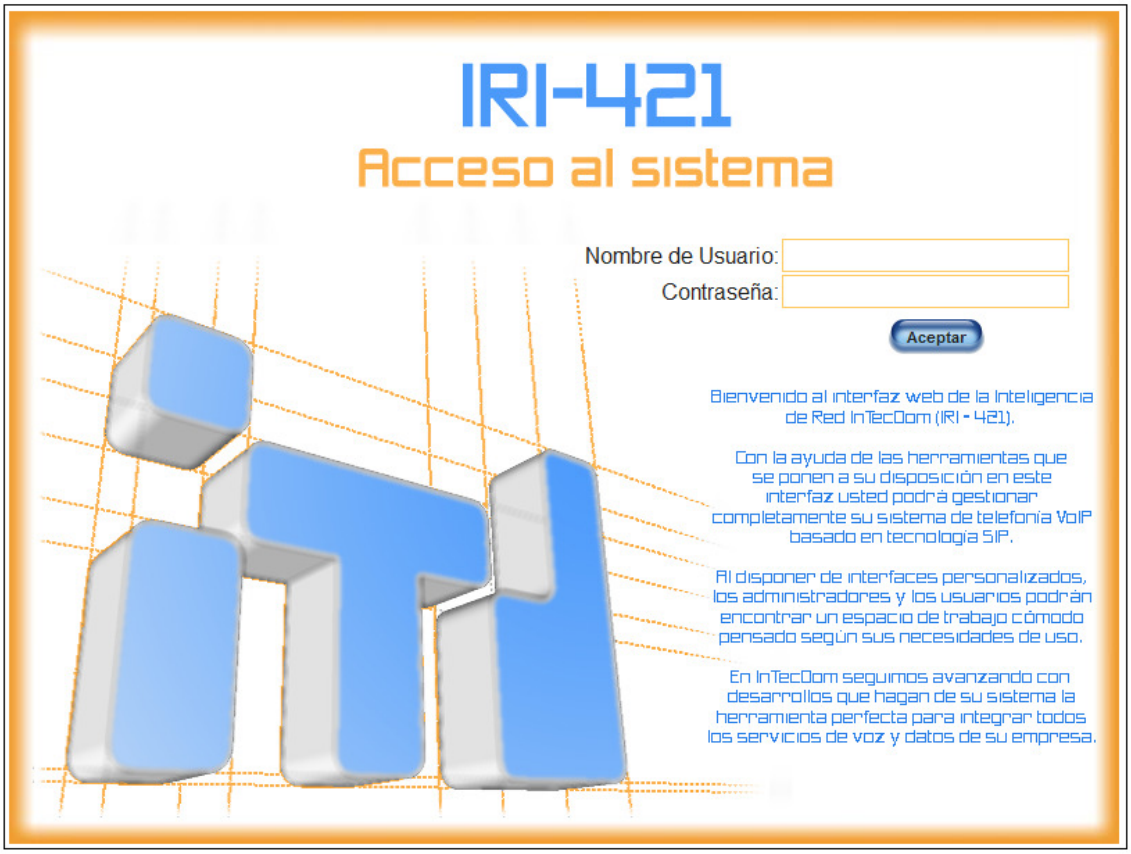

Figura 4.1: Interfaz de acceso al producto IRI-421.

La página de inicio y las vistas a las que tienen acceso un usuario con perfil Operador está compuesta por las vistas de un Usuario general de la IRI, más las vistas de un usuario Operador.

Una vez realizada la autenticación correcta del usuario operador se muestra la interfaz que aparece en la Figura 4.2.

Figura 4.2: Interfaz de inicio de IRI-421.

Un usuario con perfil Operador disfruta de los permisos de un Usuario general de la inteligencia de red, pudiendo hacer uso de las funcionalidades básicas del perfil Usuario, como modificar su agenda o acceder al directorio o al menú de "Mi usuario" cuya utilización se describe en el manual de *IRI*. Además de las funcionalidades del perfil Usuario tendrá acceso a las utilidades propias del perfil Operador, mediante el interfaz de operador y el menú de configuración del mismo que serán descritas a continuación.

4.3. Vistas de Operador.

En el menú superior de la página de inicio, se muestran las vistas de usuario:

- Configuración: Pantalla desde la que se permite realizar la personalización del interfaz de operador.
- Interfaz de Operador: Pantalla que se encarga de proporcionar al usuario la información actualizada sobre el estado de la centralita.
- Mi Usuario: Interfaz que permite al usuario editar algunos de los datos registrados en la centralita.
- Directorio: Pantalla que muestra todos los usuarios registrados en la centralita.

- Agenda: Interfaz que permite al usuario almacenar registros telefónicos de los contactos que le resulten relevantes.

4.3.1. Interfaz de Operador.

En esta sección se describen las funcionalidades ofrecidas por el "Interfaz de Operador".

Para acceder a dicha funcionalidad, se debe seleccionar la pestaña de "Interfaz de Operador" que aparece en el menú de la parte superior de la página de inicio, tal y como se muestra en la Figura 4.3.

Figura 4.3: Selección de la pestaña del Interfaz de Operador.

Una vez seleccionada la pestaña, si las prestaciones del equipo utilizado cumplen ciertas restricciones, se accede a la interfaz de operador que ofrece un aspecto como el representado en la Figura 4.4.

Figura 4.4: Vista general del interfaz de operador.

Se puede observar que el interfaz se divide en cuatro partes bien diferenciadas.

- Extensiones monitorizadas.
- Agenda del usuario operador.
- Llamadas en curso del sistema.
- Líneas analógicas y digitales del sistema.

Cada una de estas divisiones se describe detalladamente en los siguientes apartados.

4.3.2. Extensiones monitorizadas.

En el cuadrante superior izquierdo se muestran las extensiones disponibles en el sistema. Las extensiones que se visualizan y su distribución se pueden seleccionar mediante la configuración del terminal de operador que se aborda más adelante en el presente manual.

En la Figura 4.5 se puede visualizar un ejemplo del cuadrante encargado de la monitorización de las extensiones del sistema. Se puede observar que el sistema dispone de la extensión del usuario "operador" y otros usuarios *SIP*.

La información que se muestra en cada extensión es el identificador del usuario y del número de extensión asignado a éste.

Figura 4.5: Vista general de las extensiones monitorizadas.

En esta pantalla se monitoriza en todo momento el estado de las extensiones. Se representan cinco posibles estados de las extensiones:

1. En el caso de que el usuario *SIP* no se encuentre correctamente registrado o el terminal telefónico del usuario se desconecte de la red o de la alimentación, la extensión se mostrará en color gris, denotando que no está disponible. Apareciendo la imagen representada en la Figura 4.6.

Figura 4.6: Extensión con estado no disponible.

2. En el caso de que el usuario esté correctamente registrado y no esté involucrado en ninguna llamada, su extensión aparece en color verde denotando que se encuentra libre y preparado para la operación. La imagen correspondiente se representa en la Figura 4.7.

Figura 4.7: Extensión con estado libre.

3. Si el terminal de usuario está sonando porque es el destinatario de una llamada, su extensión aparecerá en color amarillo, tal y como se muestra en la Figura 4.8.

Figura 4.8: Extensión con estado sonando.

4. En el caso en que el usuario se encuentra ocupado en una conversación, tanto si la ha iniciado él como si es el destinatario de la misma, su extensión aparece en color rojo y se muestra sobre la misma superficie, la información sobre quién es su interlocutor.

   En el caso de que el interlocutor sea otro usuario del sistema, se muestra su identificador o su inicial y el apellido, en función de la elección realizada en la

configuración del terminal. En la Figura 4.9 se aprecia un ejemplo del estado de una extensión del sistema ocupada en una llamada interna.

Figura 4.9: Estado ocupado llamada interna.

En el caso en que se trate de una llamada desde o hacia exterior, se mostrará el nombre correspondiente, si el número implicado está registrado en la agenda como se puede ver en la Figura 4.10.

Figura 4.10: Estado ocupado con interlocutor registrado.

O si se trata de un número de teléfono no registrado en la agenda, se muestra directamente el número involucrado, tal y como se puede observar en la Figura 4.11.

Figura 4.11: Estado ocupado con interlocutor no registrado.

Hay que señalar, que debido a la limitación de espacio que presenta la imagen empleada para la representación de las extensiones, en el caso de que se supere el espacio reservado para la representación del interlocutor, se colocan puntos suspensivos y se recorta el nombre o el número de teléfono hasta el tamaño necesario.

5. Por último, en caso de que todas las llamadas del usuario en cuestión se encuentren retenidas, su extensión correspondiente refleja la situación mediante el color naranja, como se puede ver en la Figura 4.12.

Figura 4.12: Estado del usuario con llamadas retenidas.

A parte de mostrar el estado de las extensiones en todo momento, la subpantalla reservada a la monitorización de las extensiones ofrece la posibilidad de establecer una llamada entre el usuario operador y cualquiera de las extensiones disponibles.

Si el usuario encargado de la monitorización se coloca con el ratón, sobre la representación de la extensión deseada y realiza un doble click, el efecto que se consigue consiste en que comienza a sonar el terminal telefónico del operador y en el momento en el que éste sea descolgado, se inicia el establecimiento de una llamada con la extensión seleccionada.

El procedimiento descrito se puede visualizar en la Figura 4.13.

Figura 4.13: Inicio de una llamada interna mediante el terminal.

4.3.3. Agenda del usuario operador.

En el cuadrante inferior izquierdo se muestra la agenda del usuario "Operador" con el que se ha accedido al sistema. Se dispone de la información del nombre, apellidos y del número de teléfono o el número de la extensión en su defecto, aunque no tiene mucho sentido introducir usuarios del sistema dentro de la agenda, se ofrece la posibilidad de hacerlo. La forma de introducir contactos dentro de la agenda se describe en uno de los apartados siguientes. En la Figura 4.14 se puede apreciar un ejemplo de la pantalla correspondiente a la agenda.

Figura 4.14: Vista general de la agenda.

En la parte inferior de la subpantalla aparece un cuadro en el cual se puede introducir un número no disponible en la agenda y realizar una llamada pulsando el botón con la etiqueta "Marcar". Dicho cuadro también sirve para realizar una búsqueda dentro de la agenda, filtrando los elementos que correspondan con el patrón buscado.

En la Figura 4.15 se muestra un ejemplo de uso. El usuario ha tecleado "mov" en el cuadro reservado a tal efecto y como resultado se visualizan los teléfonos móviles guardados en la agenda, ya que corresponden a las entradas que contienen el patrón introducido.

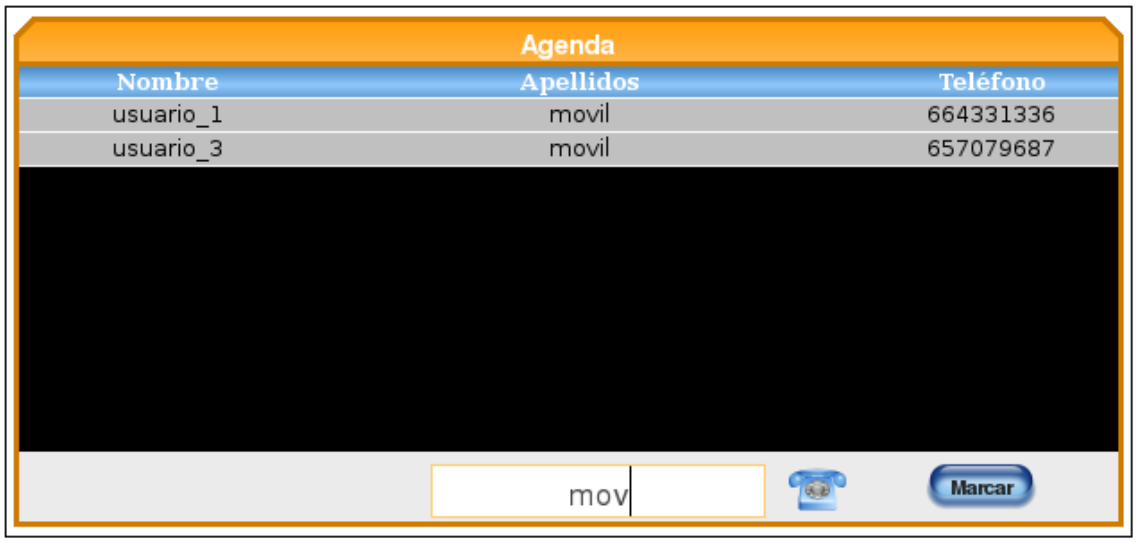

Figura 4.15: Vista de ejemplo de búsqueda de un patrón en la agenda.

La pantalla reservada a la agenda del usuario también permite establecer una llamada mediante un doble click. Para ello, el usuario operador debe colocarse con el ratón sobre la entrada de la agenda que corresponda con el número de teléfono con el que desea establecer la comunicación. Con el fin de diferenciar el contacto escogido éste se presenta con la fuente aumentada. Una vez seguro de su elección, basta con realizar el doble click sobre el contacto para que primero suene el terminal telefónico del operador y en el momento en el que éste descuelgue, se inicie el establecimiento de una llamada con el número de teléfono seleccionado.

En el apartado dedicado a las llamadas del sistema se describe el procedimiento que permite transferir tanto las llamadas entrantes como las salientes directamente hacia un número de teléfono registrado en la agenda.

En el apartado dedicado a las líneas analógicas y digitales del sistema se describe el procedimiento de establecimiento de una llamada con un número de teléfono registrado en la agenda, por una línea de salida en concreto.

4.3.4. Llamadas en curso del sistema.

El cuadrante superior derecho se reserva para el registro de las llamadas en curso del sistema. Las llamadas entrantes se representan sobre un fondo gris oscuro y las salientes sobre un tono amarillo pastel. De ambos tipos de llamada se muestra la información del origen, del destino y de la duración. En la Figura 4.16. Se puede observar un ejemplo de monitorización en el que aparece el registro de una llamada entrante y otra saliente.

Es importante tener en cuenta, que debido a las decisiones de diseño de la aplicación, se considera que las llamadas internas no disponen del mismo nivel de relevancia que las entrantes y salientes. Por lo tanto, el estado de las llamadas internas no aparece reflejado en la presente pantalla, sino que su evolución se puede seguir mediante la observación de la pantalla dedicada a las extensiones monitorizadas del sistema.

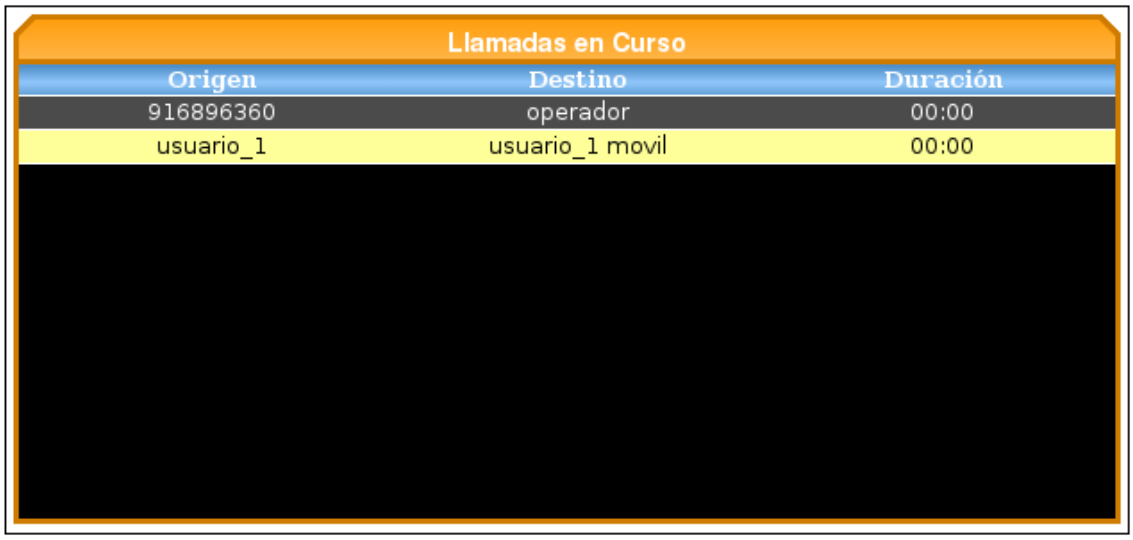

Figura 4.16: Ejemplo de llamada entrante y saliente.

El contador de la duración de la llamada se pone en marcha una vez realizada la conexión con el destino. En la Figura 4.17 se puede apreciar que la llamada entrante monitorizada, cuyo destinatario es el usuario cuyo identificador es "usuario_2", lleva establecida desde hace 21 segundos.

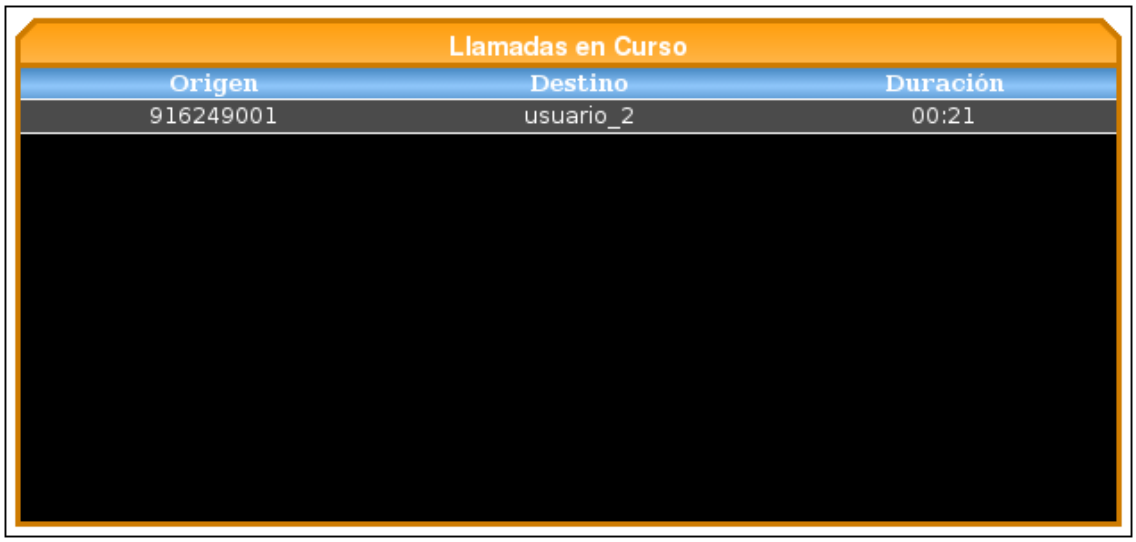

Figura 4.17: Ejemplo duración de la llamada.

En el caso de haber varias llamadas entrantes compitiendo, el operador puede elegir qué llamada quiere atender simplemente realizado un doble click sobre la llamada entrante deseada.

Es importante resaltar que una vez seleccionada una llamada por el procedimiento descrito, las llamadas restantes estarán aparcadas y cuando el operador termine de atender esa llamada a la que se le ha dado prioridad, su terminal físico ya no sonará a pesar de tener llamadas en espera. En la Figura 4.18 se representa la situación en la que el usuario operador ha dado prioridad a una de sus llamadas entrantes.

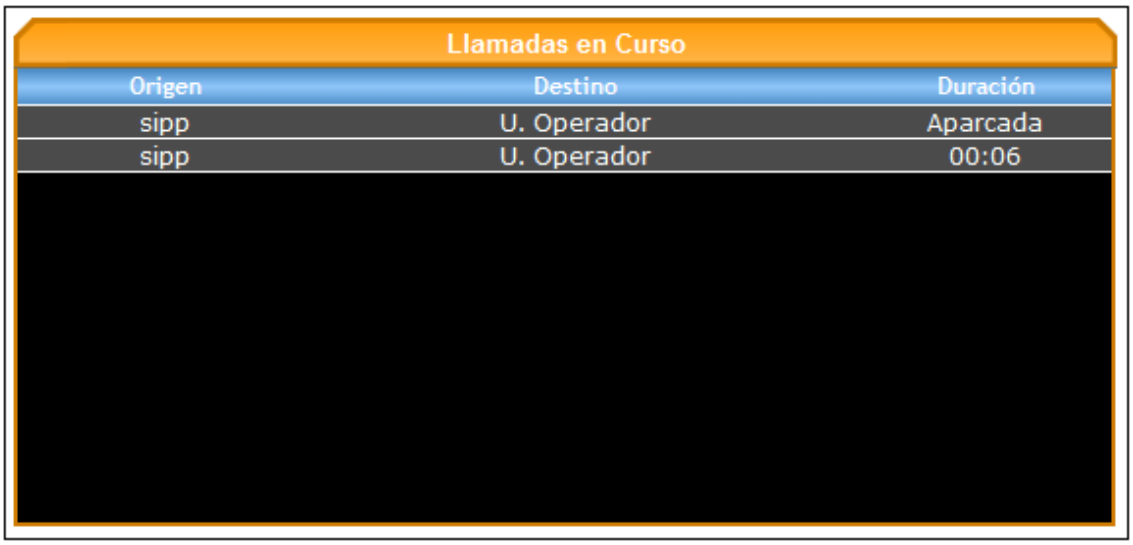

Figura 4.18: Ejemplo de priorización de una llamada entrante.

Para seguir atendiendo las llamadas, simplemente debe volver a realizar un doble click sobre la llamada a la que considere en ese momento más prioritaria.

El usuario encargado de la monitorización puede transferir las llamadas a cualquier otro usuario del sistema. El procedimiento para conseguir tal término consiste simplemente en colocarse con el ratón sobre el número de teléfono origen de la llamada y presionando el botón izquierdo arrastrarlo hacia la extensión del usuario al que se desea transferir. Para asegurarse de que se está seleccionando la llamada deseada, al colocarse con el ratón sobre éstas, aumentan de tamaño. En la Figura 4.19 se muestra un ejemplo de dicho comportamiento.

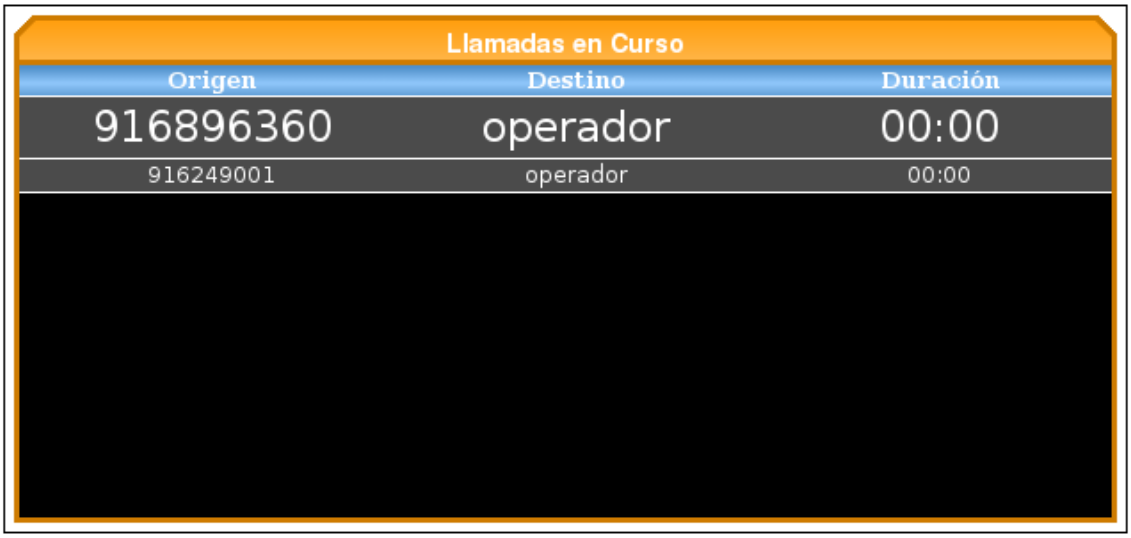

Figura 4.19: Ejemplo selección usuario.

La extensión del usuario preseleccionado se muestra en color amarillo y una vez que el operador está seguro de su selección, simplemente debe soltar el botón del ratón iniciándose la nueva comunicación. El número de teléfono origen de la llamada afectada se muestra en color rojo de nuevo para remarcar al usuario la llamada que está siendo afectada en la operación.

En la Figura 4.20 se observa la situación descrita. Al sistema está entrando una llamada y el operador desea transferirla directamente a la extensión del usuario cuyo identificador es “usuario_1”.

Figura 4.20: Inicio de un intento de transferencia de una llamada entrante a un usuario del sistema.

Como resultado del procedimiento descrito empieza a sonar el terminal telefónico del usuario seleccionado, quedando el operador libre. La situación descrita se muestra en la Figura 4.21.

Figura 4.21: Resultado de un intento de transferencia de una llamada entrante a un usuario del sistema.

Se puede apreciar que tras la maniobra descrita, el destinatario de la llamada entrante es el seleccionado usuario “usuario_1”, que es el usuario al que ha traspasado la llamada el operador, tal y como se muestra en la Figura 4.22.

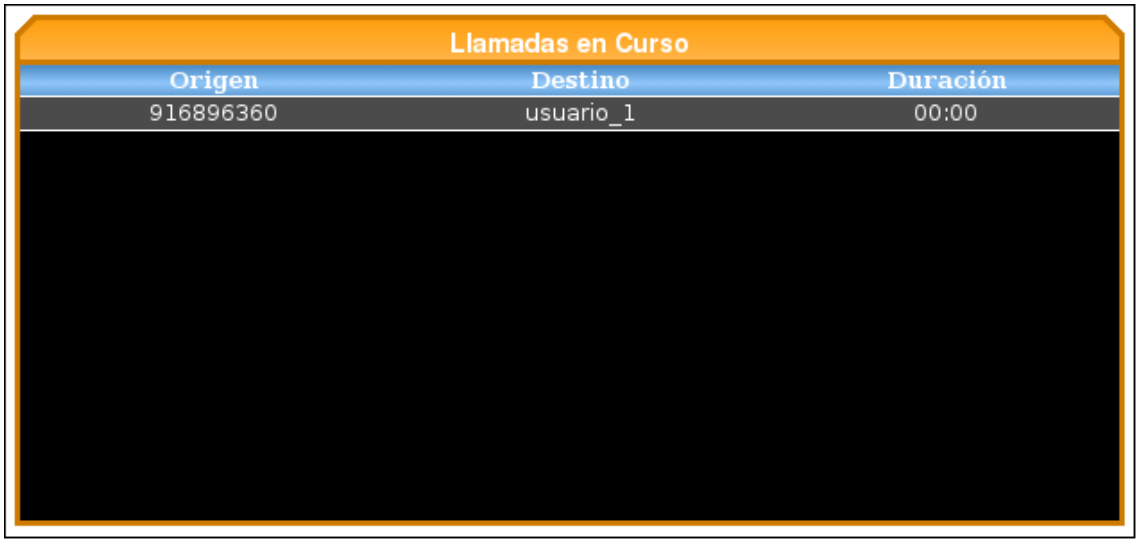

Figura 4.22: Llamada entrante transferida a un usuario del sistema.

También existe la posibilidad de traspasar la llamada entrante directamente a un número registrado en la agenda.

El procedimiento es exactamente igual que en el caso anterior. Se selecciona con el ratón el número origen de la llamada y se arrastra hasta el número de la agenda al que se desea transferir. Una vez elegido el destinatario, simplemente se suelta el botón del ratón. En la Figura 4.23 se muestra un ejemplo de la situación descrita.

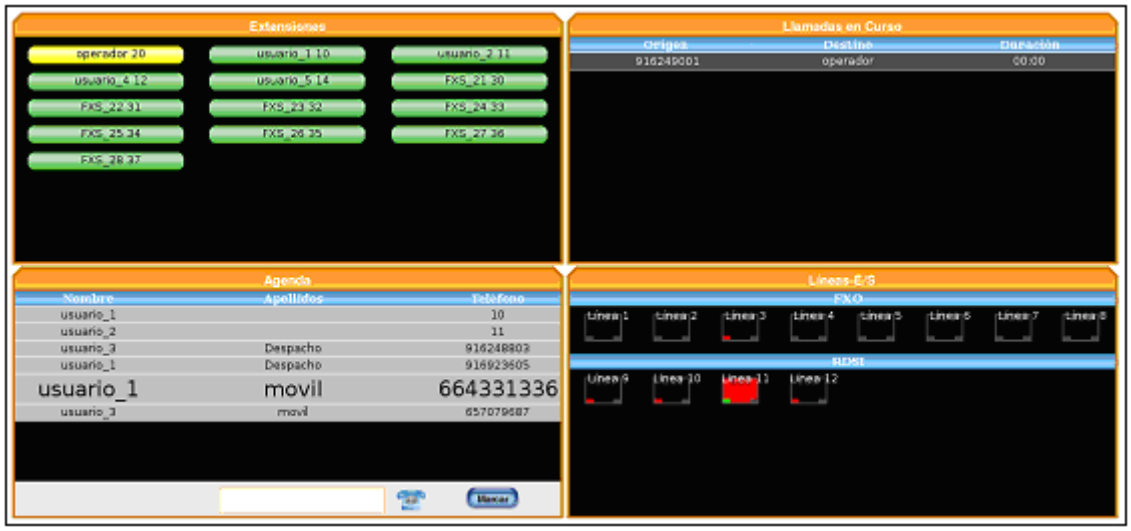

Figura 4.23: Intento de una transferencia de una llamada entrante a un contacto.

Como se puede observar, el destinatario preseleccionado se muestra de forma distintiva aumentando su tamaño. A continuación se muestra la imagen correspondiente a la situación una vez realizado el traspaso, que se puede apreciar en la Figura 4.24.

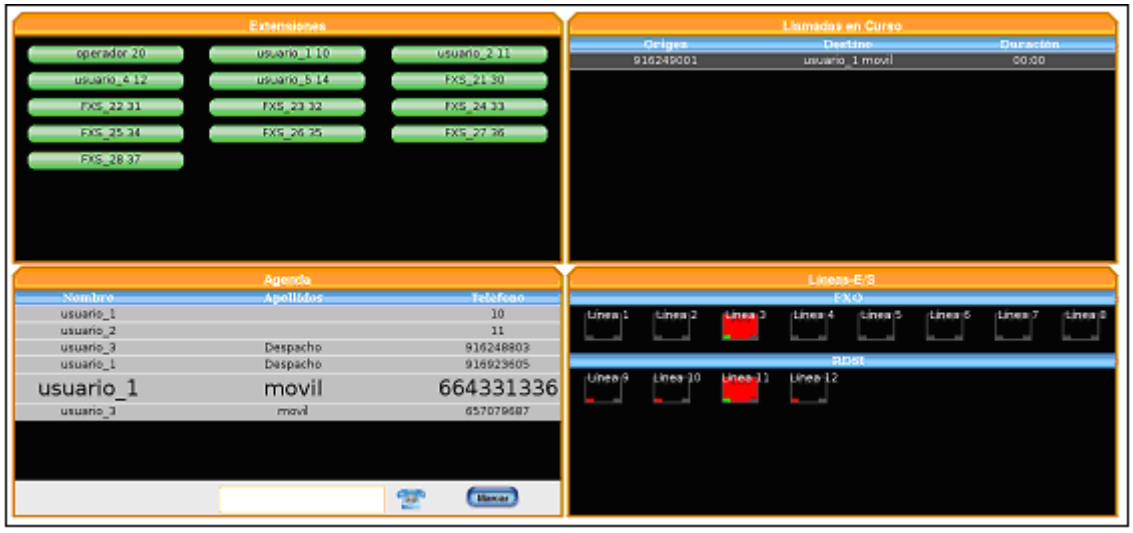

Figura 4.24: Resultado de una transferencia de una llamada entrante a un contacto de la agenda.

4.3.5. Líneas analógicas y digitales del sistema.

En el cuadrante inferior derecho se muestra la información de las líneas analógicas y digitales configuradas en el sistema. En la Figura 4.25 se puede observar que el sistema dispone de 8 líneas *FXO* de las cuales sólo la línea 3 tiene conexión física, lo cual se representa con el led rojo en la parte inferior izquierda de la imagen.

También se muestran las dos líneas BRI (*Basic Rate Interface*) del sistema, cada una con sus dos canales correspondientes.

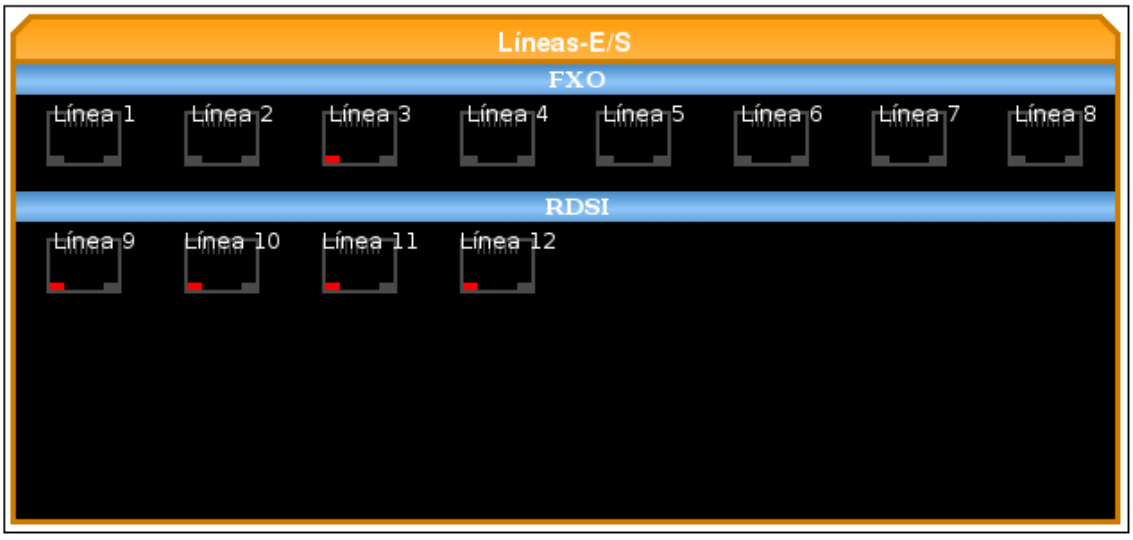

Figura 4.25: Vista general de las líneas del sistema.

El terminal de operador dispone también de la funcionalidad de detectar un corte físico en las líneas analógicas y digitales. Si por alguna razón de desconecta algún cable de las líneas *FXO*, el correspondiente terminal aparecerá con el led apagado. Si se desconecta un cable de la *RDSI*, sus dos líneas correspondientes aparecerán con los leds apagados.

En el momento en el que se genere una llamada entrante o saliente la línea que se esté utilizando en dicha comunicación aparecerá como ocupada mediante el color rojo. En la Figura

4.26 se puede observar que en el sistema hay una llamada entrante y dos salientes y las tres líneas empeladas aparecen como ocupadas.

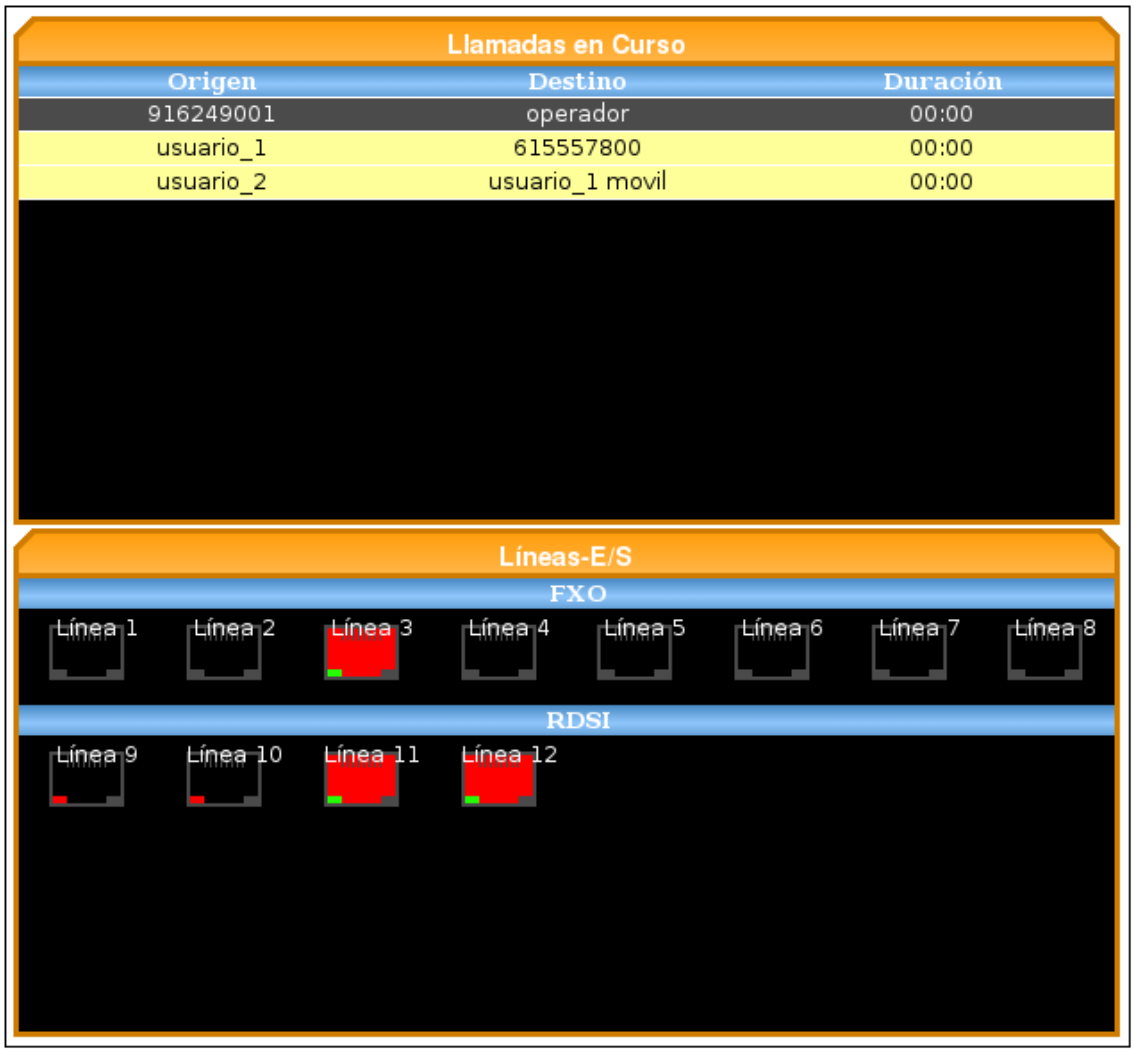

Figura 4.26: Ejemplo de ocupación de los canales.

El terminal de operador permite la selección de una línea determinada para la realización de una llamada. Dicha funcionalidad dispone de dos variantes.

- Si el número con el que se desea establecer la llamada está disponible en la agenda del operador.

Al situar el ratón sobre un número de la agenda, éste aumenta su tamaño para asegurar al usuario que está seleccionando el número correcto.

La forma de proceder en este caso consiste simplemente en posicionarse sobre el número de teléfono seleccionado, apretar el botón izquierdo del ratón y arrastrarlo hasta la línea deseada. La línea preseleccionada se muestra en color naranja como se puede observar en la Figura 4.27 y en el momento en el que el usuario esté seguro de haber seleccionado la línea deseada debe soltar el botón izquierdo del ratón.

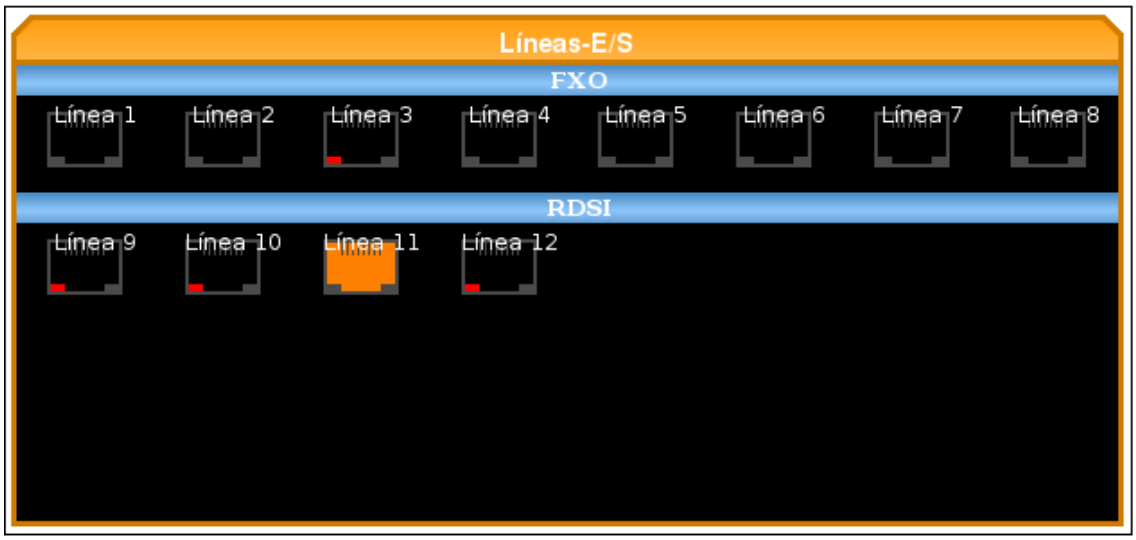

Figura 4.27: Ejemplo de un intento de encaminamiento de una llamada.

En ese momento empieza a sonar el teléfono asignado al usuario Operador y una vez éste descuelgue, se inicia el establecimiento de la llamada con el número seleccionado.

En la Figura 4.28 se puede observar el resultado del procedimiento descrito. Se aprecia en el cuadrante reservado a las llamadas del sistema, el establecimiento de una llamada saliente desde el terminal del usuario operador hacia el número de teléfono registrado en la agenda como "usuario_1 movil". A su vez, en el cuadrante reservado a las extensiones, se aprecia que el usuario operador se encuentra ocupado en una conversación con el usuario mencionado. Y por último, en el cuadrante de las líneas del sistema se puede observar que finalmente se escogió la línea 3 *FXO* para el establecimiento de la presente llamada.

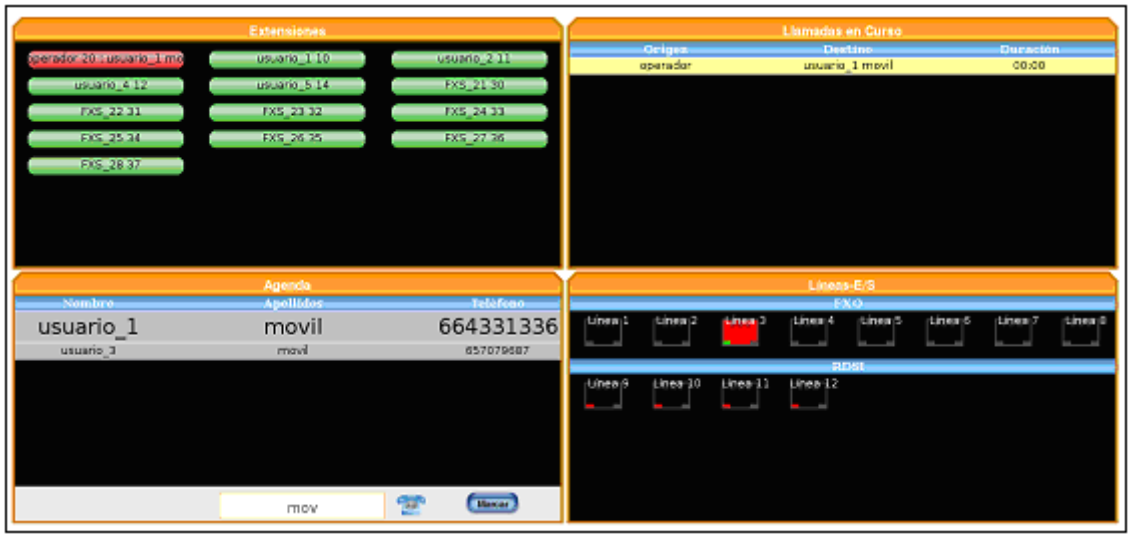

Figura 4.28: Ejemplo del resultado de un intento de encaminamiento.

- Si el número con el que se desea establecer la comunicación no está disponible en la agenda. En este caso el usuario debe introducir el número en el cuadro dispuesto a tal efecto en la pantalla de la agenda que tiene el icono de un teléfono a su derecha, cuya representación en detalle se observa en la Figura 4.29.

Figura 4.29: Espacio reservado para la introducción de datos.

A continuación, el usuario debe colocar el ratón sobre el icono de teléfono y presionando el botón izquierdo del ratón arrastrarlo hasta alcanzar la línea de salida deseada. Al igual que en el caso anterior, la línea preseleccionada se muestra en color naranja y una vez que se suelta el botón del ratón, suena el teléfono del operador. Una vez descolgado el auricular se inicia la llamada deseada. En la Figura 4.30 se muestra el resultado del procedimiento.

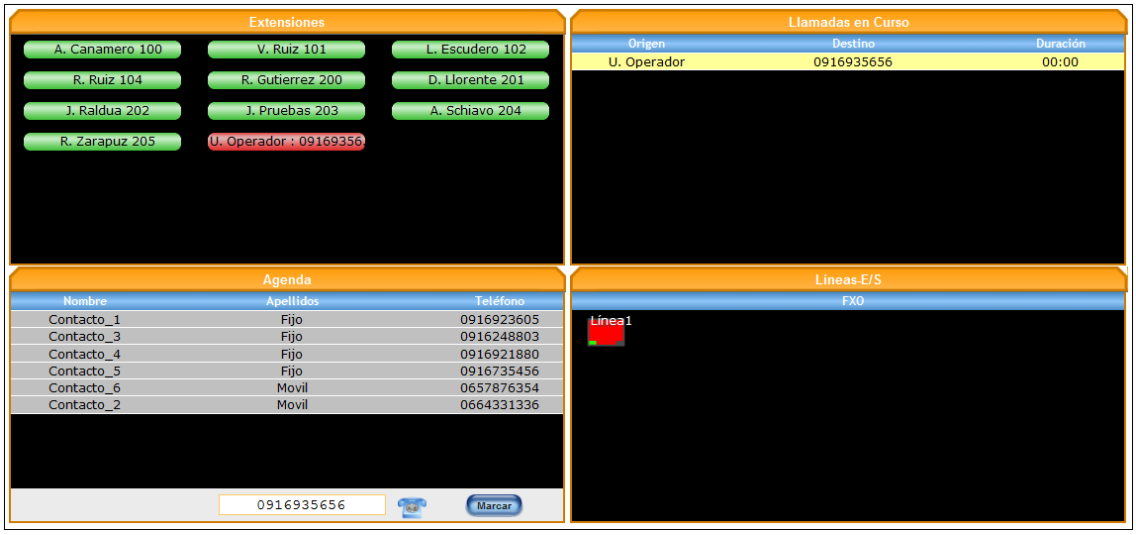

Figura 4.30: Ejemplo de encaminamiento de una llamada por una línea en concreto.

El siguiente paso es describir las funcionalidades disponibles en lo que se refiere a la configuración del terminal de operador.

4.4. Configuración del Terminal de Operador.

En esta sección se describe el procedimiento que ha de llevarse a cabo para realizar la configuración de la herramienta del interfaz de operador.

La configuración del terminal permite al usuario adaptar la herramienta lo máximo posible, dentro de unas limitaciones, a su forma de trabajo. El propósito reside en facilitar la labor del usuario encargado de la monitorización del sistema.

La aplicación dispone de una configuración por defecto, por lo tanto, el procedimiento descrito en este apartado no es obligatorio. En caso de que el usuario todavía no ha realizado una configuración de su terminal o no desea hacerlo en absoluto, la configuración por defecto le proporciona la posibilidad de visualizar las cuatro subpantallas definidas y efectuar la monitorización de todas las extensiones disponibles en el sistema a tal efecto.

Para acceder a la pantalla de configuración, desde la página de inicio, en el menú superior, se debe seleccionar la pestaña de “Configuración” tal y como se muestra en la Figura 4.31.

Figura 4.31: Selección de la pestaña de Configuración.

Seleccionando dicha pestaña se muestra el interfaz correspondiente a la Figura 4.33 que permite al usuario Operador configurar su terminal de trabajo.

La situación presentada en la Figura 4.33 corresponde a la primera vez que el usuario se dispone a configurar su interfaz, es decir, no existe registrada ninguna configuración anterior. En dicho caso, se presenta al usuario la tabla denominada "Configuración del Terminal de operador" y su parte izquierda, se muestran todas las extensiones disponibles en el sistema y sensibles a la monitorización.

Con la herramienta de las flechas representada en la Figura 4.32, que aparece entre las "Extensiones disponibles" y las "Extensiones seleccionadas", el usuario puede seleccionar las extensiones que desea que formen parte de la colección de las extensiones monitorizadas.

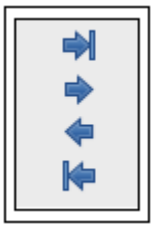

Figura 4.32: Herramienta de selección de extensiones.

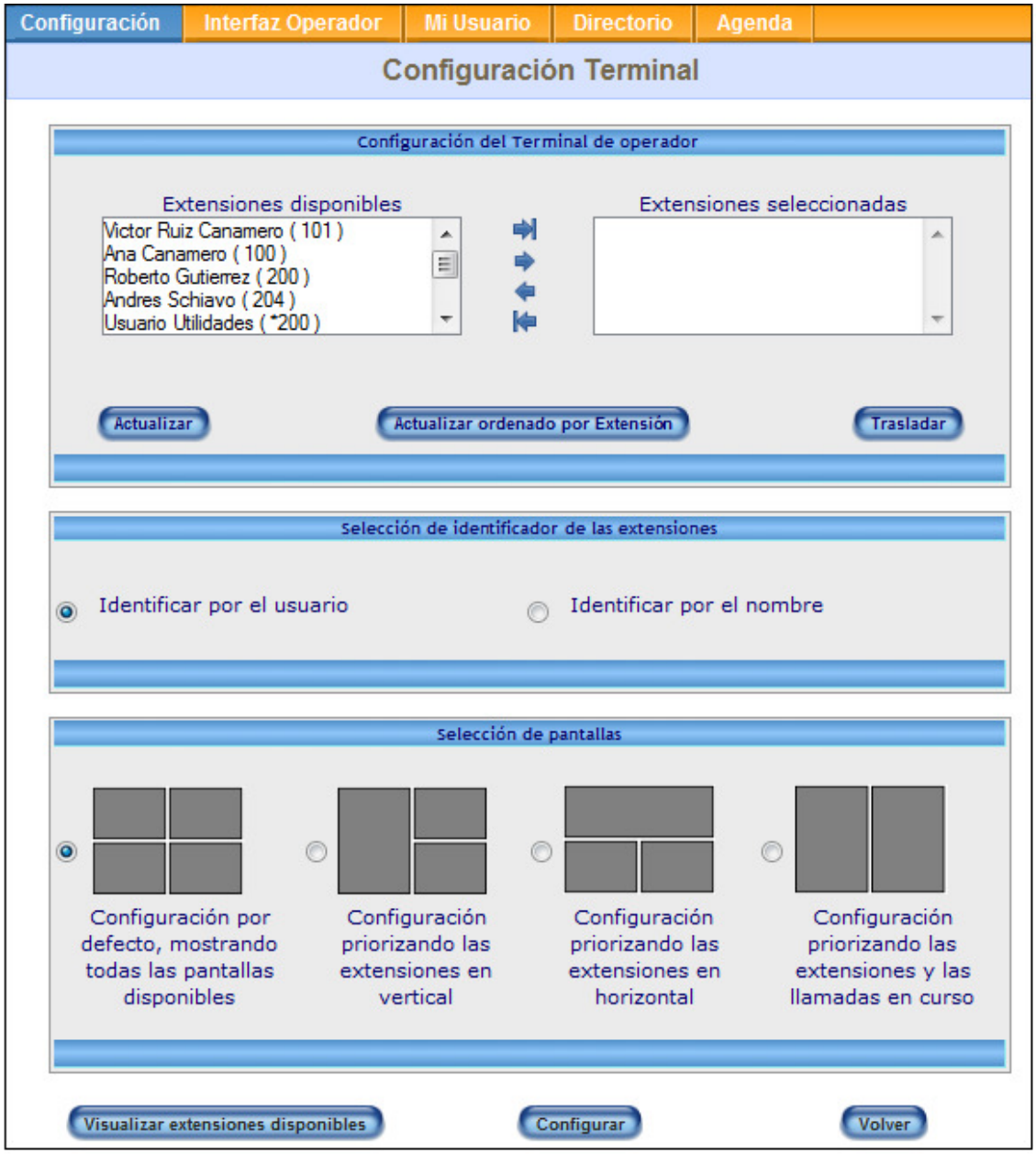

Figura 4.33: Vista general del interfaz de configuración.

Una vez que en el cuadro de las "Extensiones seleccionadas" ya están todas las extensiones que el usuario Operador desea monitorizar, el siguiente paso consiste en actualizar la tabla denominada "Panel de extensiones". Para dicha labor, existen dos posibilidades, la primera consiste en trasladar las extensiones en el mismo orden en el que aparecen en la sección "Extensiones seleccionadas". Este resultado se consigue presionando el botón de "Actualizar". La segunda opción consiste en realizar el traslado ordenando los usuarios por el número de extensión que tienen asignado, para lo cual sólo hay que presionar el botón denominado "Actualizar ordenado por extensión".

El resultado del procedimiento descrito se puede visualizar en la Figura 4.34. Como se puede ver, en esta ocasión se empleó la opción de simplemente actualizar, ya que los usuarios no aparecen ordenados por el número de extensión.

También se aprecia en la Figura 4.33 la nueva tabla, denominada "Distribución de extensiones" que dispone de la misma cantidad de posiciones que el número de extensiones seleccionadas para ser monitorizadas, encontrándose en un primer instante todas libres.

Figura 4.34: Resultado de la actualización del panel de extensiones.

La realización de la configuración de la posición de las extensiones dentro del panel de monitorización se puede llevar a cabo de dos formas.

La primera consiste en conseguir que la distribución sea idéntica a la presentada en la tabla “Panel de extensiones”. Lo cual se puede conseguir de forma automática y para ello basta con presionar el botón de “Trasladar”, obteniéndose como resultado la configuración mostrada en la Figura 4.35.

Figura 4.35: Resultado del uso del botón Trasladar.

La segunda opción consiste en realizar la configuración de la distribución de forma manual.

El procedimiento consiste en posicionarse sobre el nombre del usuario y apretar el botón izquierdo del ratón. A continuación se arrastra el nombre hasta la posición que se desea que se ocupe en el panel del Terminal del Operador. Durante el procedimiento de asignación de una posición concreta, el nombre aparece en color rojo para distinguir más claramente al usuario afectado por la configuración. Una vez asignada la posición deseada en la tabla de "Distribución de extensiones", el nombre correspondiente aparecerá en color verde en la tabla "Panel de extensiones" señalando que ese usuario ya está configurado. Para deshacer la asignación realizada basta con hacer doble click sobre el nombre en la tabla "Distribución de extensiones" y éste retornará a su lugar original. En la Figura 4.36 se puede observar un ejemplo del comportamiento descrito. Concretamente se refleja la situación en la que se ha escogido una posición determinada para la extensión 202 y se está buscando el emplazamiento más adecuado para la extensión con el número 203.

Figura 4.36: Ejemplo de un intento de configuración de la extensión de un usuario.

Mediante uno de los procedimientos descritos se realiza el posicionamiento de todas las extensiones que desea el usuario encargado de la monitorización. No es obligatorio emplear en la configuración todas las extensiones disponibles en la tabla "Panel de extensiones", sin embargo, es importante tener en cuenta que no se permite dejar huecos con la etiqueta "Libre" entre usuarios asignados, sino que los espacios sin asignar, si los hubiera, deben estar siempre al final. El resultado de una posible configuración se muestra en la Figura 4.37.

Figura 4.37: Ejemplo de una configuración manual.

El siguiente paso consiste en establecer cómo desea el usuario operador que se identifique a los usuarios monitorizados en su panel de extensiones. Para ello se dispone de la tabla llamada "Selección de identificador de extensiones" que como se puede observar en la Figura 4.38 ofrece dos posibilidades. La opción de identificar por el usuario consiste en representar las extensiones mediante su identificador de *login*.

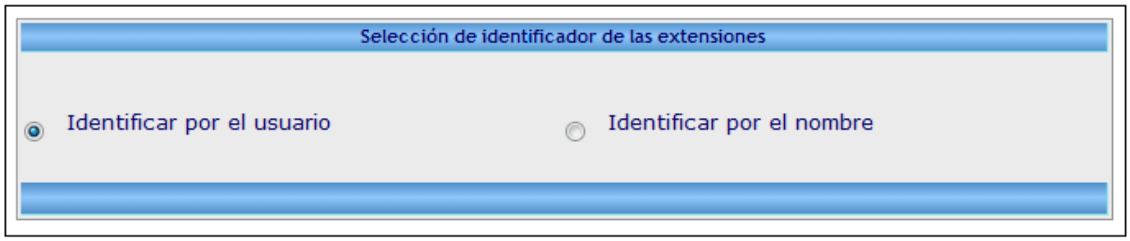

Figura 4.38: Tabla de selección de identificador de las extensiones.

En caso de seleccionar la opción descrita, el panel de extensiones que se le muestra al usuario operador durante la monitorización tiene el aspecto reflejado en la Figura 4.39.

Figura 4.39: Ejemplo de uso del login de usuario como identificador.

La otra posibilidad consiste en identificar a los usuarios mediante la inicial de su nombre seguida de su primer apellido, ofreciendo el panel de extensiones un aspecto como el registrado en la Figura 4.40.

Figura 4.40: Ejemplo de uso del nombre como identificador de usuario.

La tabla denominada “Selección de pantallas”, que aparece a continuación, permite al usuario seleccionar cómo desea que se distribuya el espacio del interfaz de operador y se puede observar en la Figura 4.41.

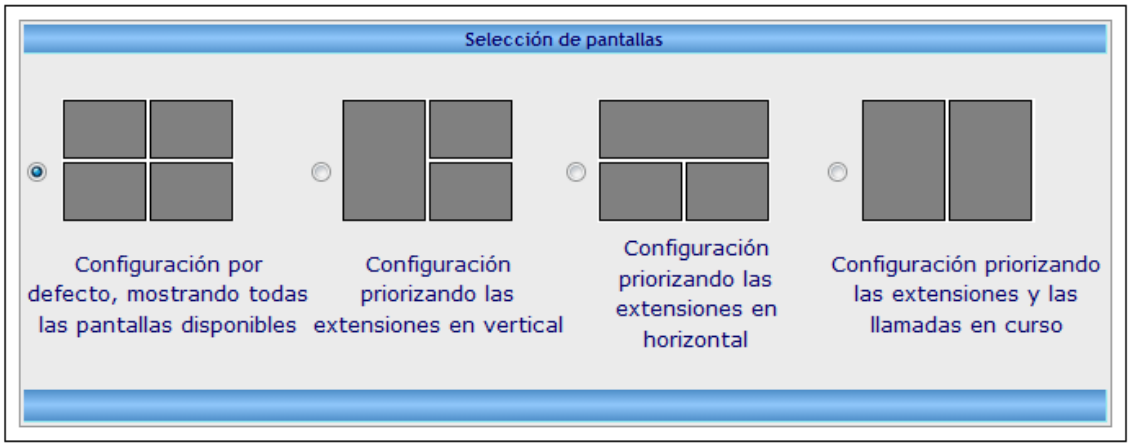

Figura 4.41: Tabla de selección de pantallas.

Existen cuatro configuraciones posibles. La primera corresponde a la configuración por defecto, en la que se muestran al usuario operador las cuatro pantallas descritas anteriormente en el manual. El aspecto del terminal de operador correspondiente se puede visualizar en la Figura 4.42.

Figura 4.42: Vista de la configuración por defecto.

La segunda configuración posible elimina la pantalla correspondiente a las líneas analógicas y digitales y asigna un mayor espacio a las extensiones del sistema. Obteniéndose un aspecto como el registrado en la Figura 4.43.

Figura 4.43: Vista de la configuración priorizando las extensiones en la vertical.

La tercera configuración es igual que la anterior a diferencia que en esta ocasión, las extensiones se muestran en la parte superior en lugar de a la izquierda y a una razón de seis por fila, en lugar de tres. El aspecto que presenta el terminal de operador en esta ocasión es el que se puede observar en la Figura 4.44.

Figura 4.44: Vista de la configuración priorizando las extensiones en la horizontal.

La última configuración posible elimina también la agenda y sólo se muestran las extensiones y las llamadas en curso del sistema, ocupando cada una la mitad del espacio disponible en la pantalla.

Una vez que se han colocado las extensiones en las posiciones deseadas, se ha escogido la forma de representar a los usuarios y se ha seleccionado la configuración de las pantallas deseada se está en disposición de consolidar la configuración realizada. Para ello basta con presionar el botón denominado “Configurar” que aparece al final de la pantalla y se puede visualizar en la Figura 4.33.

Como ya se comentó, no se permite dejar posiciones sin asignar en el medio del panel de configuración. Para controlarlo, en el caso de que se diera el caso, se muestra al usuario un

mensaje advirtiendo de la situación y no se consolidará la configuración realizada sino que se volverá a la pantalla de configuración para que se solucione el problema.

El usuario dispone en todo momento de abandonar la tarea de la realización de la configuración sin efecto alguno, apretando el botón de "Volver". También está en disposición de añadir o quitar extensiones de la configuración mediante el empleo del botón "Visualizar extensiones disponibles". La distribución de los botones descritos se puede observar en la Figura 4.45.

Figura 4.45: Vista de los botones.

4.5. Agenda.

Por último se presenta este apartado para realizar una breve descripción del procedimiento a seguir para introducir contactos en la agenda de usuario. La funcionalidad de la agenda de usuario no forma parte propiamente dicha del proyecto del terminal de operador pero para poder visualizar los contactos del usuario en el interfaz del operador, es necesario haberlos registrado previamente y por ello se ha incluido el presente apartado en el manual de usuario.

Para acceder a la pantalla de edición de la agenda, el usuario debe escoger la pestaña "Agenda" que puede visualizar en el menú de la página principal como se puede observar en la Figura 4.46.

Figura 4.46: Selección de la pestaña de agenda.

Una vez seleccionada la correspondiente pestaña, al usuario se le presenta la interfaz reflejada en la Figura 4.47. Como se puede observar, la aplicación le muestra al usuario todos los contactos que éste tiene registrados hasta el momento. Para llevar a cabo la introducción de un nuevo contacto basta con presionar con el ratón la pestaña "Nuevo Contacto" que se puede apreciar en la parte izquierda de la pantalla. La interfaz de creación de un nuevo contacto se muestra en la Figura 4.48.

Para crear la nueva entrada deseada basta con rellenar los campos que el usuario estime necesarios y presionar el botón de "Aceptar".

Hay que tener en cuenta el plan de numeración que se use en la organización donde vaya a emplearse el terminal de operador. Es decir, muchas empresas diferencian con un prefijo las llamadas hacia el exterior. Para conseguir el correcto funcionamiento del terminal es preciso crear los contactos teniendo en cuenta si se emplea dicho prefijo o no. Por ejemplo, en la Figura 4.47 se puede observar que todos los números de teléfono registrados comienzan con un cero, lo cual se debe precisamente a lo explicado antes.

Todos los contactos creados pueden volver a ser editados en el momento en el que el usuario lo desee o directamente eliminados del sistema haciendo uso de las Acciones disponibles.

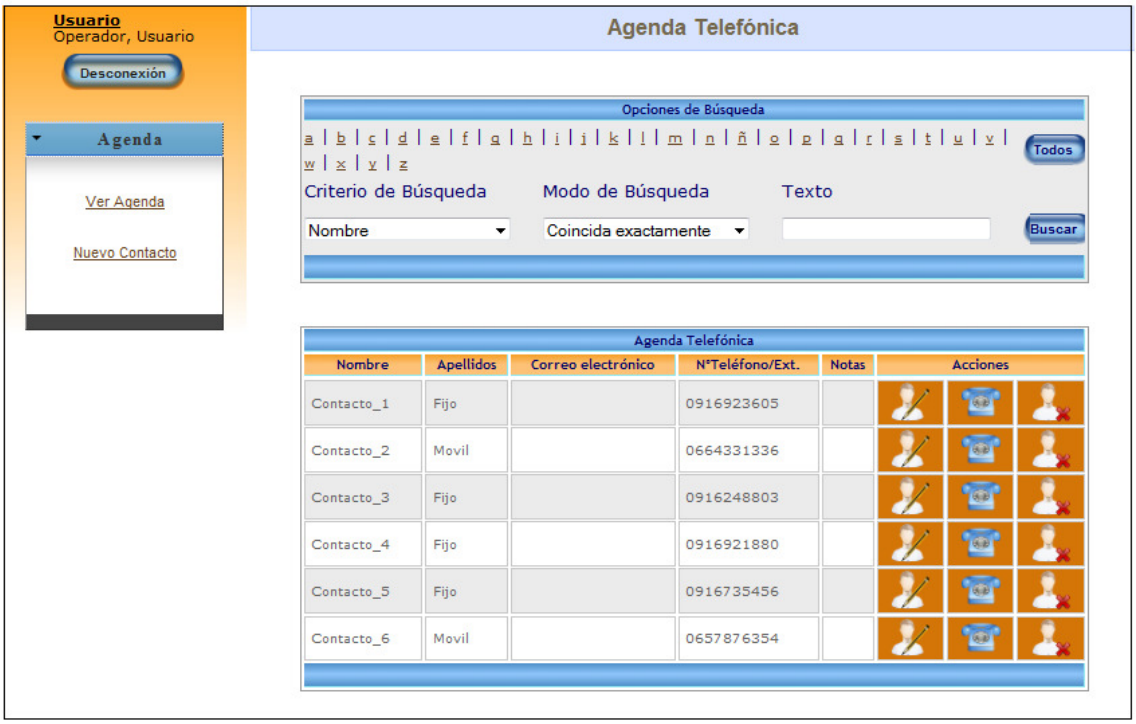

Figura 4.47: Interfaz de visualización de la agenda de usuario.

Figura 4.48: Interfaz de creación de una nueva entrada en la agenda de usuario.

CAPÍTULO 5:
MEMORIA ECONÓMICA Y PRESUPUESTO

5. MEMORIA ECONÓMICA Y PRESUPUESTO

5.1. Introducción.

La intención de este capítulo es realizar una estimación del coste real de la puesta en marcha del proyecto descrito. Para efectuar dicha estimación se han tenido en cuenta los distintos tipos de gastos en los que se incurre a la hora de realizar un proyecto. Se ha computado por ejemplo, la contribución de los gastos derivados directamente del material utilizado o los gastos debidos al personal involucrado. A su vez se han tenido en cuenta otro tipo de gastos, los originados por el alquiler de la oficina, el desembolso provocado por el uso de la conexión de ADSL, las líneas telefónicas contratadas por la empresa o incluso los gastos de luz.

5.2. Coste de materiales.

Una parte importante del coste total del desarrollo de un proyecto la comprende el gasto en el material necesario para llevarlo a cabo. En este apartado se analizará y se computará el coste tanto del software como del hardware empleado.

5.2.1. Coste del software empleado.

Debido a que el proyecto se desarrolló en una plataforma Unix y empleando diversas herramientas de licencia *GPL*, el coste debido al software empleado en realidad sólo deriva de la elaboración de la presente memoria ya que se ha optado por utilizar Microsoft Office del sistema operativo Windows. En la Tabla 1 se recoge la relación de los costes en los que se incurrió debido a la necesidad del software.

Concepto	*Cantidad*	*Precio unitario (€)*	*Importe(€)*
Licencia Microsoft Windows XP SP2	1	103	103
Microsoft Office 2007	1	130	130
Microsoft Visio 2007	1	130	130

Tabla 1: Coste del software.

5.2.2. Coste del hardware empleado.

A continuación se analiza el coste que ha supuesto la adquisición del equipo necesario para llevar a cabo el proyecto. En la Tabla 2 se presenta la relación de los gastos debidos a la adquisición del hardware empleado.

Concepto	*Cantidad*	*Precio unitario (€)*	*Importe(€)*
Portátil Hacer	1	500	500
Ordenador Core Duo, 2.4GHz	1	500	500
Xorcom Astribank 8 FXO	1	556	556
Xorcom Astribank 8 FXS	1	489	489
Xorcom Astribank 2 BRI	1	454	454
Thomson ST2030	3	90	270
Linksys IP Phone SPA942	1	95	95
Grandstream BT201	2	40	80
DI-Voz 50	2	80	160

Tabla 2: Coste del hardware.

Una vez estudiados todos los gastos en material que se produjeron a lo largo de la evolución del desarrollo del proyecto, podemos resumir los resultados en la Tabla 3.

Concepto	***Importe(€)***
Material software	363
Material hardware	3104
Total	3467

Tabla 3: Resumen de los costes de material

5.3. Coste de personal.

Con el propósito de obtener una estimación del coste de personal asociado a este proyecto, se ha realizado una identificación de las principales tareas dentro de la ejecución de dicho proyecto. Es importante tener presente que los espacios de tiempo presentados son una mera estimación debido a que numerosas tareas realizadas no fueron ejecutadas secuencialmente. Una vez determinadas la lista de tareas, se efectuó la asignación de éstas al tipo de personal correspondiente. Todo el trabajo realizado a la largo de la ejecución del proyecto se ha clasificado según la cualificación que debía presentar su ejecutor, es decir, las tareas se han asignado o a un ingeniero superior, o a uno técnico o a un administrativo. El esfuerzo requerido para cada tarea se evaluó sobre los supuestos de una dedicación del 100%, en una jornada de 8 horas diarias trabajando 20 días al mes.

Para realizar el cálculo del gasto que suponen los recursos humanos para la realización del presente proyecto se ha utilizado el dato del coste de una hora de ingeniero que la empresa en la que se ha realizado el proyecto, cobra a sus clientes, y que es 35€/hora en horario de oficina, para el caso del ingeniero superior. Para el caso del ingeniero técnico, se han empleado los honorarios medios de un diplomado utilizados en los contratos realizados por la Universidad Carlos III de Madrid, que ascienden a unos 22€/hora. Y por último, para el trabajo reservado al personal administrativo, se consideró que el coste es de unos 18€/hora.

A continuación se describe la clasificación de las tareas realizadas a lo largo del proyecto y se adjuntan las tablas correspondientes que recogen el tiempo invertido en llevarlas a cabo y el coste en el que se incurre.

- Tareas ingeniero superior:

1. Investigación previa. El producto cuyo desarrollo se narra en la presente memoria fue ideado como complemento del producto principal de la empresa *InTecDom*. Una vez observada la oportunidad presente en el mercado de una posible necesidad de los clientes de la empresa de adquirir una herramienta que permitiera la monitorización e interacción con su centralita, se realizó una investigación sobre productos similares existentes en el mercado y se buscó una forma de diferenciación. A su vez, se investigó sobre las tecnologías disponibles para llevar a cabo la implementación.

2. Definición del producto. Tras la recapitulación de toda la información se estudió la demanda real y se efectuó la definición de la especificación del producto.

3. Estudio de *Asterisk*. Se realizaron consultas con expertos en *Asterisk* para asegurar tanto la viabilidad del producto como para averiguar la mejor forma de enfrentarse al diseño de éste.

4. Diseño del producto. Una vez que se disponía de toda la información y requisitos necesarios se efectuó el diseño del producto a alto nivel, se acordaron las tecnologías a utilizar y se propusieron posibles herramientas para llevarlo a cabo. Tras lo cual se pasó el testigo a un ingeniero técnico encargado de la implementación del sistema.

Las tareas descritas se recogen en la Tabla 4 junto con las horas que se estima que se han empleado para llevarlas a cabo.

Concepto	***Tiempo(horas)***	***Importe(€)***
Investigación previa	80	2990
Definición del producto	20	748
Estudio de *Asterisk*	40	1495
Diseño del producto	50	1750

Tabla 4: Resumen de costes de las tareas de un ingeniero superior.

- Tareas ingeniero técnico:

1. Estudio de *API AsteriskJava*: Fue necesario un breve estudio de las posibilidades que ofrecía dicha librería. A su vez, se fue aprendiendo la forma de utilizarla

2. Implementación del código: Tras la familiarización con las herramientas que serían empleadas en la implementación y teniendo claro el diseño del producto, se dispuso a la elaboración del código que daría vida a la idea propuesta.

3. Desarrollo de la interfaz: Esta tarea se realizó de forma concurrente con la anterior. Consistió en diseñar y elaborar la interfaz web que serviría de nexo entre el usuario encargado de la monitorización y el sistema. Esta labor comprendió tanto diseño gráfico como la implementación de la comunicación con la parte principal del programa.

4. Realización de las pruebas: Esta tarea presentó un peso importante en el conjunto total del proyecto debido a la naturaleza del mismo. Hubo que asegurarse de que ante cualquier tipo de actividad de la centralita el interfaz era capaz de reflejar fielmente la situación.

5. Puesta en marcha: Una vez completada la etapa de las pruebas del correcto funcionamiento de la herramienta, se implantó en el sistema de la empresa donde se realizó para seguir observando su comportamiento como etapa final antes de la implantación como producto definitivo.

Concepto	***Tiempo(horas)***	***Importe(€)***
Estudio *API AsteriskJava*	30	630
Implementación código	720	15120
Desarrollo de la interfaz	60	1260
Realización de las pruebas	160	3360
Puesta en marcha	20	420

Tabla 5: Resumen de costes de las tareas de un ingeniero técnico.

- Tareas personal administrativo:

La tarea de realizar la documentación completa del proyecto realizado fue encomendada al personal administrativo. En la Tabla 6 se presenta el coste estimado de las labores administrativas realizadas en el presente proyecto.

Concepto	***Tiempo(horas)***	***Importe(€)***
Documentación	150	2700

Tabla 6: Resumen de costes de las tareas de un administrativo.

Una vez obtenidos los importes correspondientes a cada tipo de personal involucrado en las tareas, se puede calcular el coste de personal total incurrido para llevar a cabo el presente proyecto y la cifra resulta ser de 29373€.

5.4. Otros gastos.

En este apartado se incluye el resto de los gastos en los que se incurre a lo largo de la realización de un proyecto de este estilo. Se realizado un cómputo de los gastos debidos al alquiler de la oficina de la empresa implicada, los gastos derivados de las líneas telefónicas utilizadas, la línea *ADSL*, el material de oficina y los costes de luz, calefacción y aire acondicionado. Se ha considerado que dichos gastos suponen la parte proporcional al personal involucrado en el proyecto del total incurrido por la empresa a lo largo del tiempo empleado para el desarrollo de éste. El coste obtenido asciende a 1100 €.

5.5. Coste total de ejecución.

Finalmente, tras realizar los cálculos pertinentes de todos los gastos necesarios para el desarrollo del presente proyecto, ya se está en disposición de obtener el coste total de ejecución que simplemente consiste en la suma de los gastos totales en material y los costes del personal necesarios. A continuación se muestra la tabla resumiendo dichos resultados:

Concepto	***Importe(€)***
Gasto en materiales	3467
Coste personal	29373
Otros gastos	1100
Total	33940

Tabla 7: Resumen coste de ejecución.

A la cifra obtenida hay que añadir el Impuesto de valor añadido (*IVA*) que durante el período de la ejecución del proyecto ascendía al 16%, consiguiendo un valor de 39370.4€

Con lo cual, podemos concluir diciendo que la estimación del coste del proyecto es de ***treinta y nueve mil trescientos setenta euros con cuarenta céntimos***.

Julia Koblents Lápteva

Febrero 2010

CAPÍTULO 6:
CONCLUSIONES Y POSIBLES MEJORAS

6. CONCLUSIONES Y POSIBLES MEJORAS

Este capítulo se dedica a elaborar un análisis del trabajo realizado dentro del contexto descrito en el capítulo del Estado del Arte y obtener las conclusiones relevantes sobre la utilidad real del producto obtenido. Se pretende también discutir si el producto final cumple con las características propuestas inicialmente y si las funcionalidades de éste son suficientes para su utilización en un escenario real. También se procura relatar las dificultades y problemas encontrados a lo largo del desarrollo del proyecto y comentar las medidas tomadas para su solución.

6.1. Conclusiones.

Como se ha comentado en el capítulo de Estado del Arte, *Asterisk* está obteniendo cada vez más adeptos por las numerosas prestaciones que ofrece y que siguen creciendo gracias a los desarrollos que continúan realizando los usuarios actuales de *Asterisk*. A pesar de la existencia de alternativas perfectamente válidas, es *Asterisk* quien consigue atraer un mayor número de seguidores. Debido a que dichos seguidores se encargan también de realizar mejoras y añadir nuevas funcionalidades, *Asterisk* se está convirtiendo en una plataforma cada vez más potente, lo cual a su vez favorece de nuevo su mayor expansión.

La presente memoria describe la realización de una herramienta que supone un complemento para una centralita basada en *Asterisk*. Es importante tener en cuenta las características especiales de este proyecto por formar parte de un esquema mucho mayor. Es decir, la mayor parte de las herramientas y tecnologías a utilizar estaban impuestas debido a esta circunstancia, siendo por tanto incoherente analizar la idoneidad de la elección de éstas. Sin embargo, es importante resaltar que se ha conseguido cumplir los objetivos propuestos y se ha obtenido un producto final con salida en el mercado.

La herramienta ofrece una completa monitorización del sistema, mostrando el estado actualizado de los componentes seleccionados en tiempo real. A su vez, permite al usuario operador interactuar con la centralita enviando órdenes a través del interfaz.

Se consiguió realizar un interfaz sencillo y de uso intuitivo de manera que con un elemental manual de usuario, los administradores de los sistemas son capaces de realizar su labor sin necesidad de conocer el funcionamiento real de *Asterisk*, que era uno de los objetivos principales de este proyecto.

Como se ha explicado en el capítulo introductorio, la realización del proyecto se programó mediante diferentes fases. Una de las fases donde se encontraron más dificultades fue en la definición del producto. La dificultad residía en que para establecer las funcionalidades que sería capaz de ofrecer la herramienta era imprescindible cierto nivel de conocimiento de *Asterisk* para ser capaz de discernir las opciones que podrían llevarse a buen término y las que no. A parte de eso, había que encontrar la forma correcta de enviar la orden a la centralita mediante el uso de la librería que se propuso para tal fin. Es decir, también hubo que estudiar a fondo las posibilidades de ésta y ver de qué forma se podía adaptar a lo que se pretendía conseguir.

Una de las fases finales del desarrollo consistió en la elaboración de una batería de pruebas para cerciorarse de que el producto cumplía con las especificaciones determinadas al principio del proceso. Pero en realidad es importante destacar la importancia que tuvieron las pruebas a lo largo de todo el proceso de la implementación. Hubo que enfrentarse a un problema muy importante que se pudo detectar gracias a las pruebas realizadas como complemento del trabajo. Se observó que los eventos que generaba el manager de *Asterisk* ante algunas situaciones variaban con la configuración de la centralita. Debido al hecho de que la

información sobre el estado del sistema se obtiene mediante el análisis de los eventos generados por manager de *Asterisk*, había que tener en cuenta que la colección de éstos dependía de la configuración.

Otro punto importante a considerar fue tener controlado el consumo de recursos del sistema para llevar a cabo la monitorización. Se pudo comprobar que el producto realizado no sobrecargaba el sistema en ningún momento ya que tanto el uso de la CPU como de memoria era reducido incluso trabajando a plena carga.

La herramienta desarrollada está diseñada para ser utilizada como un instrumento de monitorización de las centralitas en pequeñas y medianas empresas. Sin embargo, bajo la definición de empresa mediana se encuentran organizaciones de hasta 250 trabajadores. Como se ha comentado ya en varias ocasiones a lo largo de esta memoria, la limitación del espacio de representación en la pantalla de usuario encargado de la monitorización ha sido un factor muy importante a lo largo de todo el desarrollo.

Una vez conocida la herramienta, resulta evidente que no tiene sentido emplearla para monitorizar la actividad de un número tan elevado de usuarios porque sencillamente el observador no podrá sacar nada en claro con ese exceso de información. Sin embargo, no deja de cumplir con su propósito ya que incluso en las empresas con un número de trabajadores tan elevado, la cantidad de usuarios reales de una centralita es mucho menor. Además, no todos ellos presentan interés para una posible monitorización, como se pudo concluir tras el estudio de mercado realizado por la empresa en la etapa inicial del proyecto. Por lo tanto, el grupo total se reduce a un número máximo aproximado de 50 trabajadores, que es perfectamente asumible para la herramienta construida.

Se puede concluir por tanto, que el producto obtenido está orientado a su uso en pequeñas y medianas empresas. Aunque, tal vez, en compañías con un número de usuarios muy elevado, sería más interesante y productivo disponer de varios operadores pero siempre por la comodidad del usuario y no por las limitaciones de la herramienta.

Como ya se comentó a lo largo de esta memoria, en el momento de comenzar el desarrollo del presente producto, ya existían herramientas con funcionalidades similares. No obstante, el propósito era desarrollar un interfaz integrado y que cumpliera requisitos muy específicos. Además, se procuró suplir las deficiencias de las herramientas disponibles en el mercado, estudiando en profundidad tanto las posibilidades ofrecidas por *Asterisk* como las demandas de los clientes. Como resultado se ha obtenido un software idóneo para el uso conjunto con la centralita desarrollada por la empresa *InTecDom*, siendo consciente que en el caso de utilizar por ejemplo, un simple *Asterisk*, sin los servicios complementarios ni sin demandas especiales, existen herramientas de código libre bastante completas y seguramente más acertadas en ese caso de uso.

6.2. Posibles mejoras.

Como ya se ha mencionado, tras el trabajo llevado a cabo, se puede concluir que se ha conseguido la implementación de una herramienta con las funcionalidades establecidas en la especificación, que cumple los objetivos propuestos y que satisface las demandas de los clientes hacia los cuales estuvo enfocado su diseño desde el comienzo del proyecto. Sin embargo, como cualquier producto es susceptible de mejorar y como se pretende que sea una herramienta viva y que responda de la forma más adecuada a las necesidades de los usuarios, se han identificado algunas posibles puntos a mejorar.

- Se comenzará comentando la posibilidad de mejorar el interfaz con respecto a su presentación gráfica. Para la realización de la herramienta se empleó un *CSS* básico, y en su diseño no participó ningún experto en las artes gráficas. El objetivo principal era conseguir una herramienta funcional, dejando su presentación para un segundo plano. Es por ello, que

se propone como una posible mejora, realizar un diseño más profesional y más adecuado a las características del producto.

- Como complemento a la parte de la representación, podría resultar interesante adaptar el producto para su uso con una pantalla táctil. Sería necesario estudiar cuál sería el tamaño de pantalla adecuado y si el incremento en su precio pudiera llegar a ser aceptado por los posibles clientes.

- En el apartado dedicado a la descripción de las herramientas utilizadas se explicó que para la implementación de los efectos visuales empleados en la interfaz, se recurrió a una librería llamada *Script.aculo.us*. A lo largo de la ejecución del trabajo, se detectaron algunos problemas derivados del comportamiento definido de los elementos disponibles. Sin embargo, como no supuso la imposibilidad de llevar a cabo de ningún proceso deseado y para conseguir cumplir con la programación de la ejecución del proyecto, no se recurrió al empleo de otra librería. Pero una vez concluido el trabajo, se puede considerar como una posible mejora utilizar una librería que se adapte más a las necesidades del producto, teniendo también en cuenta que *Script.aculo.us* es una librería mucho más pesada que otras disponibles.

- Un posible trabajo futuro sería estudiar el comportamiento de la herramienta al enfrentarse a la monitorización de centralitas con un número de usuarios más elevado. Debido a la simple limitación del espacio disponible para la representación de la información obtenida, el producto se planteó desde el principio para la monitorización de no más de treinta extensiones. Sin embargo, la centralita dispone de un límite de usuarios mucho mayor y por lo tanto sería interesante estudiar la posibilidad y la forma de emplear la herramienta en casos más extremos. A parte de la restricción del espacio habría que estudiar si un aumento significativo de los usuarios monitorizados no supondría un excesivo aumento de consumo de recursos.

- Las centralitas *Asterisk* permiten la definición de componentes del sistema denominados colas. Dicha funcionalidad consiste en poder definir un número que cuando éste se marque, la llamada será encaminada hacia un conjunto de usuarios que están registrados como agentes de la cola correspondiente. La funcionalidad descrita permite la implementación de un *call center* y de otras funcionalidades.
  Un factor importante que podría resultar muy negativo es el hecho de que la herramienta elaborada no está preparada para procesar los eventos que se generan en una centralita que emplea colas. Dicha limitación se debe a que el diseño de la herramienta se realizó bajo un supuesto bastante concreto y no se tuvo en cuenta el caso especial de las colas.
  Por lo tanto, una mejora importante consistiría en completar la herramienta para que fuera capaz de monitorizar la actividad de las colas definidas en el sistema. Como consecuencia de la presente mejora sería necesario también introducir cambios que permitieran el tratamiento especial del que disfrutan los agentes de las colas.

- La versión del producto sobre las características de la cual se está realizando la presente memoria, es capaz de reflejar el hecho de que un usuario se encuentra participando en una sala de conferencia. Durante el proceso de la implementación, se observó que podría ser interesante permitir al usuario la organización de una sesión en una sala de conferencia desde el interfaz. Se propone por tanto como una posible mejora la posibilidad de que el usuario convoque una reunión en una sala de conferencias y que por ejemplo, los participantes sean avisados por una locución, y a la hora pactada tan sólo tengan que descolgar el auricular de su terminal si aceptan la invitación de participación.

- Como consecuencia de la observación de la herramienta a lo largo de su desarrollo, se fueron detectando diversas formas de completar su funcionalidad y en esta etapa del proyecto se proponen como posibles mejoras para versiones futuras.

  Se percibió como interesante la oportunidad de ofrecer al usuario la posibilidad de iniciar la grabación de una llamada que éste estime necesario, mediante la interacción con el terminal.

  En una etapa inicial del diseño se estudió la posibilidad de ofrecer al usuario encargado de la monitorización, la funcionalidad de mandar un correo electrónico a otro usuario del sistema. Sin embargo se descartó debido a que la herramienta se concibió como una interfaz de interacción dinámica, de presentación de la información actualizada en tiempo real y se consideró que el correo no encajaba en dicha definición.
  Tras acabar el proyecto se observa que fue una decisión acertada, no obstante se puede considerar interesante la posibilidad de enviar mensajes cortos de voz, a otros usuarios, mediante la interacción con el interfaz. Otra posibilidad sería la introducción de una especie de chat para poder intercambiar mensajes cortos con otros usuarios del sistema.

  Como se ha comentado en la memoria técnica y en el manual de usuario, la herramienta permite al encargado de la monitorización del sistema realizar diversas acciones con las llamadas. Entre dichas acciones se encuentra la posibilidad de aparcar una llamada para atender otra que se considere más prioritaria en un momento determinado. En el momento en el que se aparca una llamada, el interlocutor de ésta pasa a escuchar una música en espera. Se considera que podría resultar útil, permitir al usuario emitir órdenes de reproducir alguna locución al interlocutor que se encuentra a la espera. Por ejemplo, si éste considera que no va a poder atender la llamada en un tiempo determinado o que directamente decida interrumpir definitivamente una conversación, que exista la posibilidad de reproducir una locución para informar al interlocutor de los planes del usuario del terminal.

- Se habló en anteriores capítulos de la necesidad detectada de ofrecer la posibilidad de que el terminal de operador fuera configurable. En el momento en el que se reveló la carencia, se realizó un análisis y se propuso un modelo de configuración del terminal. Sin embargo, a lo largo de la implementación de la herramienta se observó que podría resultar interesante ofrecer un sistema de configuración más flexible y que permitiera al usuario adaptar la herramienta totalmente a su gusto. Por ello, se propone esta idea como un posible trabajo futuro para la mejora del producto.

- Se comentó en el capítulo de la memoria técnica, en el apartado en el que se describían las decisiones de diseño que al comienzo del trabajo se barajó la idea de presentar al usuario encargado de la monitorización un registro de las últimas llamadas. Como ya se explicó, dicha idea se descartó por falta de espacio físico de representación en la pantalla. Tras observar el uso de la herramienta en un escenario real, se considera interesante disponer de un registro de la actividad y se propone como un posible trabajo futuro implementar un complemento que lo permita. Debido a la limitación de espacio, no puede presentarse en el mismo interfaz pero sería útil tener acceso a una información detallada sobre las llamadas recibidas y enviadas, cambios de estado relevantes de los usuarios, llamadas no atendidas, etc.

- Se observó que la tarificación de las llamadas es un tema que preocupa mucho a los potenciales clientes del presente producto. Por lo tanto, se considera interesante ofrecer la información sobre el coste de las llamadas. Como ya se explicó, el usuario operador dispone de la posibilidad de escoger la línea por la que desea que se encamine su llamada. Dicha funcionalidad se introdujo originariamente por motivos de seguridad pero se puede emplear con el propósito de escoger una línea con la tarificación más adecuada para cada caso. Disponer de la información del coste instantáneo no parece muy relevante pero sí que se

considera interesante ofrecer un resumen de los costes por ejemplo a través del registro de actividad mencionado en el apartado anterior.

CAPÍTULO 7:
REFERENCIAS Y BIBLIOGRAFÍA

7. REFERENCIAS Y BIBLIOGRAFÍA

7.1. Glosario.

A
ACD: Automatic Call Distribution
ADF: Application Development Framework
ADSL: Asymmectric Digital Suscriber Line
AGI: Asterisk Gateway Interface
AJAM: Asyncronous Javascript Asterisk Manager
AJAX: Asynchronous Javascript And Xml
AMI: Asterisk Manager Interface
AMP: Asterisk Management Portal
API: Application Programming Interface

B
BRI: Basic Rate Interface
BSD: Berkeley Software Distribution

C
CDR: Call Detail Record
CPU: Central Processing Unit
CRLF: Carriage Return LineFeed

D
DECT: Digital Enhanced Cordless Telecommunications
DHCP: Dynamic Host Configuration Protocol
DNS: Domain Name System

E
EJB: Enterprise Java Beans

F
FastAGI: Fast Asterisk Gateway Interface
FCP: Firewall Communication Protocol
FXO: Foreign Exchange Office
FXS: Foreign Exchange Station

G
GCC: GNU Compiler Collection
GPL: General Public License

H
HTML: HyperText Markup Language
HTTP: HyperText Transfer Protocol

I
IAX2: Inter-Asterisk eXchange
IDE: Integrated Development Environment
IETF: Internet Engineering Task Force
IMAP: Internet Message Access Protocol
IP: Internet Protocol
IRI: Inteligencia de Red de InTecdom
IVR: Interactive Voice Response

J
JDBC: Java Data Base Connectivity
JDK: Java Development Kit
JMS: Java Message Service
JMX: Java Management eXtensions
JNDI: Java Naming and Directory Interface
J2EE: Java 2 Platform, Enterprise Edition
JSF: Java Server Faces
JSP: Java Server Pages

L
LGPL: Lesser General Public License

M
MGCP: Media Gateway Control Protocol
MP3: MPEG-1 Audio Layer 3MPEG-1 Audio Layer 3

P
PBX: Private Branch Exchange
PC: Personal Computer
PCM: Pulse Code Modulation
PDF: Portable Document Format
PL/SQL: Procedural Language/Structured Query Language
PoE: Power over Ethernet
PRI: Primary Rate Interface
PSTN: Public Switched Telephone Network

R
RDSI: Red Digital de Servicios Integrados
RJ45: Registered Jack 45
RSA: Rivest, Shamir and Adleman
RTP: Real time Transport Protocol

S
SAR: Service ARchive
SCCP: Skinny Client Control Protocol
SER: SIP Express Router
SIP: Session Initiation Protocol

SMS: Short Message Service
SQL: Structured Query Language
SS7: Signaling System Number 7

T
TCP: Transmission Control Protocol
TDM: Time-Division Multiplexing
TIFF: Tagged Image File Format

U
UML: Unified Modeling Language

V
VoIP: Voice over Internet Protocol
VPN: Virtual Private Network

X
XML: eXtensible Markup Language
XMPP: Extensible Messaging and Presence Protocol

Y
YATE: Yet Another Telephony Engine

7.2. Referencias.

[1] J. Davidson, J. Peters, M. Bhatia, S. Kalidindi, S. Mukherjee. *Voice over IP fundamentals.*

[2] Roger L. Freeman. *Fundamentals of telecommunications.*

[3] Charles M. Kozierok. *A TCP/IP guide.*

[4] A. Bruce Carlson. *Sistemas de comunicación: una introducción a las señales y el ruido en las comunicaciones eléctricas*.

[5] Alan V. Oppenheim. *Signals and Systems.*

[6] T. George, B. Bidulock, R. Dantu, H. Schwarzbauer, K. Morneault. Signaling System 7 (SS7). *RFC 4165. IETF.*

[7] B. Schliesser, T. Nadeau. Definition of Textual Conventions for Virtual Private Network (VPN) Management. *RFC 4265.IETF.*

[8] Charles K. Summers. *ADSL: standards, implementation and architecture.*

[9] Gary C. Kessler,Peter V. Southwick. *ISDN: concepts, facilities and services.*

[10] [Online] https://developer.skype.com/ [Consulta: enero 2010]

[11] [Online] http://www.itu.int/ [Consulta: enero 2010]

[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. SIP: Session Initiation Protocol. *RFC 3261. IETF.*

[13] M. Spencer, B. Capouch, E. Guy, F. Miller, K. Shumard. IAX: Inter-Asterisk eXchange Version 2. *RFC 5456. IETF*.

[14] [Online] http://www.asterisk.org/ [Consulta: enero 2010]

[15] Brian W. Kernighan, Dennis M. Ritchie. *The C Programming language.*

[16] [Online] http://www.linux.org/ [Consulta: enero 2010]

[17] [Online] http://www.bsd.org/ [Consulta: enero 2010]

[18] [Online] www.apple.com/macosx/ [Consulta: enero 2010]

[19] [Online] http://www.oracle.com [Consulta: enero 2010]

[20] [Online] http://www.microsoft.com/ [Consulta: enero 2010]

[21] http://www.digium.com/en/ [Consulta: enero 2010]

[22] F. Andreasen, B. Foster. Media Gateway Control Protocol (MGCP) Version 1.0. *RFC 3435. IETF.*

[23] [Online] http://www.freeswitch.org/ [Consulta: enero 2010]

[24] [Online] http://gcc.gnu.org/ [Consulta: enero 2010]

[25] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. *RFC 3920. IETF.*

[26] [Online] http://xmpp.org/extensions/xep-0166.html [Consulta: enero 2010]

[27] [Online] http://www.iptel.org/ser/ [Consulta: enero 2010]

[28] [Online] http://www.kamailio.org/ [Consulta: enero 2010]

[29] K. Egevang, P. Francis. The IP Network Address Translator (NAT). *RFC 1631. IETF.*

[30] [Online] http://www.sipfoundry.org/ [Consulta: enero 2010]

[31] [Online] http://www.cisco.com/ [Consulta: enero 2010]

[32] [Online] http://products.nortel.com/ [Consulta: enero 2010]

[33] [Online] http:// www.ietf.org/ [Consulta: enero 2010]

[34] [Online] http://www.fredshack.com/docs/yate.html [Consulta: enero 2010]

[35] [Online] http://www.gnu.org/software/bayonne/ [Consulta: enero 2010]

[36] [Online] http://www.callweaver.org/ [Consulta: enero 2010]

[37] [Online] http://www.opensips.org/ [Consulta: enero 2010]

[38] [Online] http://www.trixbox.org/ [Consulta: enero 2010]

[39] [Online] http://www.vercomsystems.com/ [Consulta: enero 2010]

[40] John Dunlop,D. Geoffrey Smith. *Telecommunication engineering*.

[41] M.Crispin. Internet Message Access Protocol Version 4 rev1. *RFC 3501. IETF*.

[42] [Online] http://www.hylafax.org/ [Consulta: enero 2010]

[43] [Online] http://www.soft-switch.org/ [Consulta: enero 2010]

[44] [Online] http://www.dect.org/ [Consulta: enero 2010]

[45] [Online] http://www.voip-info.org/ [Consulta: enero 2010]

[46] Thomas A. Powell,Fritz Schneider. *JavaScript: the complete reference.*

[47] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol HTTP/1.1. *RFC 2616. IETF.*

[48] [Online] http://www.intecdom.es [Consulta: enero 2010]

[49] Bill Burke, Richard Monson-Haefel. *Enterprise JavaBeans 3.0.*

[50] [Online] http://java.sun.com/ [Consulta: enero 2010]

[51] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 3550. IETF.*

[52] Mark Richards,Richard Monson-Haefel,David A. Chappell. *Java Message Service.*

7.3. Bibliografía.

"*Asterisk, The Future of Telephony*" 2nd Edition. **Jim Van Meggelen**, **Leif Madsen**, **Jared Smith**. Ed. O'Reilly.

"*The Asterisk Handbook*" Version 2. **Mark Spencer**, **Mack Allison**, **Christopher Rodes**, **The Asterisk Documentation Team**. Draft Digium.
http://www.digium.com/handbook-draft.pdf

Sitios web:

Sitio oficial de *Asterisk*.
http://www.asterisk.org/ [Consulta: enero 2010]

Foro sobre VoIP.
http://www.voipforo.com/ [Consulta: enero 2010]

Portal sobre VoIP.
http://www.voip-info.org/wiki-Asterisk [Consulta: enero 2010]

Tutorial de *jboss*.
http://docs.jboss.org/jbossas/jboss4guide/r1/html/index.html [Consulta: enero 2010]

Portal con tutoriales.
http://www.adictosaltrabajo.com/tutoriales/ [Consulta: enero 2010]